WEATHER FORECASTING PREDICTION

USING MACHINE LEARNING TECHNIQUES

Submitted in partial fulfilments of the

for the award of the degree of

Bachelor of computer sciences and Engineering

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
of Supervisor :
Designation

Submitted By

Name of Student/s

NIKHIL TIWARY

(19SCSE1010415)

NISHANT KUMAR

(19SCSE1010068)

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

REPORT

Table

ABSTRACT

Weather forecasting is the use of science and technology to predict the atmospheric conditions of a given area. Ancient methods of weather forecasting often relied on patterns of visual events, also called pattern recognition. For example, it can be noted that if the sun went down too bright, the next day would often bring about better weather. However, not all of these predictions prove reliable. Here the system will predict the weather according to parameters such as temperature, humidity and air. This program is a web application with the function of visual clicks. The user will log in to the system using the user ID and password. The user will enter the current temperature; moisture and air, the System will take this parameter and predict the weather from the previous data in the database. The role of the administrator is to add the previous weather data to the database, so that the system will calculate the weather based on this information. The weather forecast system takes into account parameters such as temperature, humidity and air and will forecast the weather based on the previous record so this forecast will appear to be reliable. This system can be used in Air Traffic, Marine, Agriculture, Forestry, Military, and Navy etc.

In this report, a re-enacted framework is created to foresee different climate conditions utilizing Data Analysis and Machine learning procedures, for example, straight relapse and strategic relapse. The primary wellspring of information to be utilized for directed taking in is to be gathered. The current climate condition parameters ex. temperature and so on are utilized to fit a model and further utilizing machine learning methods and extrapolating the data, the future varieties in the parameters are broke down.

Introduction

Weather forecasting is the use of science and technology to predict the atmospheric conditions of a given area. Ancient methods of weather forecasting often rely on visual event patterns, and are also called pattern recognition. For example, it can be noted that if the sun went down too bright, the next day would often bring about better weather. Here the system will predict the weather according to parameters such as temperature, humidity and air. This program is a web application with the function of visual clicks. The user will log in to the system using the user ID and password. The user will enter the current temperature; moisture and air, the System will take this parameter and predict the weather from the previous data in the database. The role of the administrator is to add the previous weather data to the database, so that the system will calculate the weather based on this information.

The weather forecast system takes into account parameters such as temperature, humidity and air and will forecast the weather based on the previous record so this forecast will appear to be reliable. This system can be used in Air Traffic, Marine, Agriculture, Forestry, Military, and Water, etc. Predicting the temperature and rainfall of a particular day and day is the main purpose of this paper. In the paper we predict the rain and heat of Europe; year to 2051 and predicts global warming; year to 2100.Our paper is intended to provide real-time weather service at the most appropriate level of granularity. We take the user's location (longitude, latitude) using the GPS data service whenever a user requests our services. Our system will process users' queries and will extract data from our repository to get relevant results. Users will also be provided with recommendations which is an important centre of our service. Personalized predictions are made for each user based on their location.

Literature Survey

Customer churn management, as part of CRM, has become a major problem. In telecommunications, the term "churn" means the loss of subscribers from one provider to another over a period of time. According to a previous study [3], the average churn of telecommunications is about 2.2% per month. This means that one in 50 subscribers of a given company discontinues their services on a monthly basis. As it is more profitable to retain existing customers than to constantly attract new customers [4] - [6], it is important to build an accurate churn predictive model to identify those customers most inclined to catch. The literature created in the customer churn uses a variety of data mining technologies, such as neural networks [7], integration [8], decision tree [7], [9], retrofitting [10], [11], vector support machine [4], [12], and then integrate integrated methods [13], to provide accurate predictions. According to a review of customer prediction models [14], Postponement is the most common method to be performed, probably because of its high definition and accuracy of understanding of major drivers, as well as providing details for setting maintenance actions.

Since churner usually takes up only a portion of the customer base, the problem of customer churn prediction is always associated with the problem of distributed segmentation or the lack of churner data. One of the most common methods of coping is the sample [15] .The methods that use the sampling process alter the distribution of training models and create balanced training settings to create predictive models of churn [8], [9], [12], [13]. However, a recent study [11] on the issue of class inequality in churn prediction revealed that the sampling process used did not increase speculative performance.

Although the proposed system of randomized forestry [11] is proposed, tree compounds, such as Random Forests, are often criticized for being difficult to interpret [16], [17], that is, difficult to detect

from abandonment, so they are not the preferred methods in this study. Other studies examine the potential for new features of fraudulent predictions, such as the social network [18] and the details of customer complaints document [19], beyond the paper.

Developed literature only uses augmentation as a standard way to strengthen accuracy, and few researchers have ever tried to take advantage of a given weight by increasing algorithms. Weight provides valuable information, especially for marketers.

TECHNOLGY

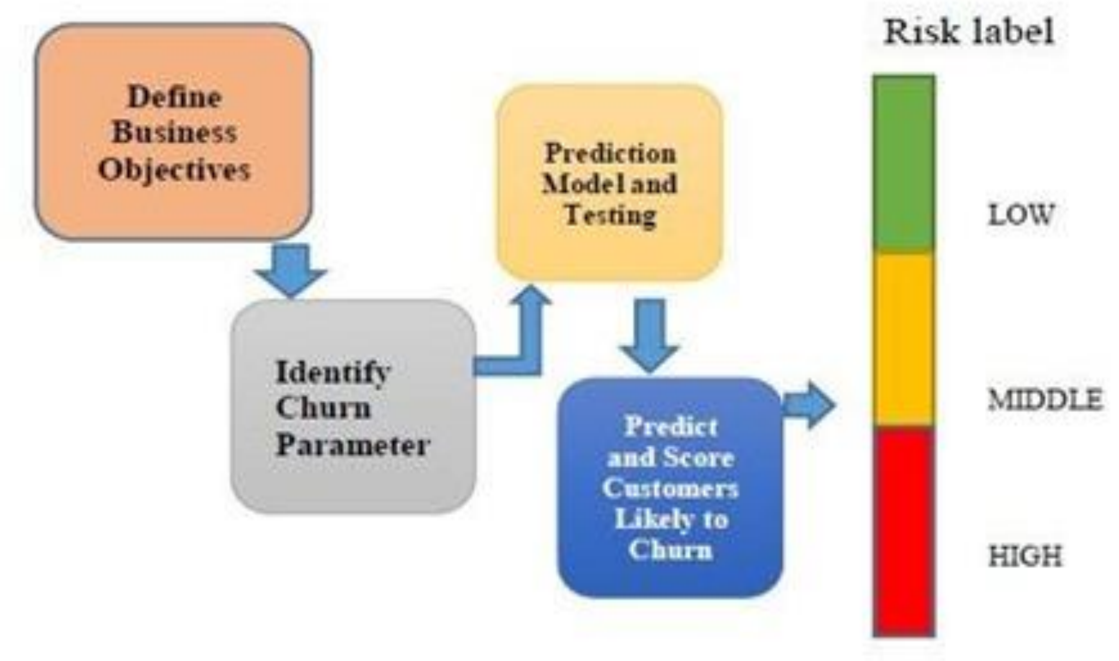

Define Business Objectives
Identify Churn Parameter
Prediction Model and Testing
Predict and Score Customers Likely to Churn
Risk label
LOW
MIDDLE
HIGH

Fig no.1: Technology Diagram

This section explains how the Ensemble based Classifiers and well-known Base Classifiers were used in the Churn Prediction Model.

A. Decision Tree

Decision Tree was developed to overcome the drawbacks of ID3 algorithm. C4.5 utilizes the benefits of greedy approach and uses a series of rules for classification. Although this approach gives a high classification accuracy rate it fails to respond to noisy data. Gain is the main metric used in the decision tree to decide the root node attribute.

B. Naive Bayes

Naïve Bayes is a brute-force method for training the model. The underlying principle behind Naïve Bayesian classifier is Bayes Theorem. For the classification problem, each predictor attribute was consider separately with class label for model construction using training dataset. Predictor attribute includes the area, service calls, evening calls night calls etc. Apply the conditional probability for each attribute belongs to all the predictor attributes given that class label represents churn. The disadvantage of this methods is, it is not suited for the large dataset.

C. Support Vector Machine

SVM algorithm was proposed by Boser, Guyon, and Vapnik. It was very well used for both classification and regression problem. SVM maps all the data points to a higher dimensional plane to make the data points linear separable. The plane which divides data points is known as hyper plane. It can be used for small dataset to give an optimal solution. SVM cannot be more effective for noisy data. SVM model tries to find out the churn and non-churn customer. In order to divide the dataset into churner and non-churner group, first it will take all the data points in n dimensional plane and divide the data points into churner and

non-churner group based on maximum marginal hyper plane.

Based on the maximum marginal hyper plane it will divide the data points into churner and non-churner group. Here n represents the number of predictor variable associated with the dataset.

D. Bagging

Bagging (or Bootstrap aggregation) is one of the Ensemble based Classifiers which consist of bag of similar type or dissimilar type base classifiers. Bagging algorithm helps to reduce the variance of the classifier used for the Churn Prediction Model in order to increase the performance. The steps in designing of Churn Prediction Model using Bagging algorithm are as follows, First, it is required to divide the input dataset into k subset with replacement, then it requires to train the model by using the (k-1) subset and test the model using the dataset which has not been used for training model. The experimental results showed that, Bagging is effective, because it predicts the test instances using the classifier which has more accuracy from the bag of classifier, Bagging requires heavier computational resource for the Model construction.

E. Boosting

Boosting Ensemble technique is designed in such a way that it will maintain a weight for each training tuple. After a classifier is learned from the training tuple, weights are updated for the subsequent classifier. The final Boosted Classifier combines the vote of each individual classifier for prediction to improve the performance of the classifier. In similar to SVM, this model is also not suited for noisy data. The key idea for the customer Churn Prediction using Boosting algorithm is to train a series of classifier simultaneously and keep updating the model accuracy for improving the performance of the classifier.

F. Random Forest

Random forest (Bierman 2001) works based on the random subspace method. The designed strategy used in Random Forest is divide and conquer. It forms number of Decision Trees and each Decision Tree is trained by selecting any random subset of attribute from the whole predictor attribute set. Each tree will grow up to maximum extent based on the attribute present in the subset. Then after, based on average or weighted average method, the final Decision Tree will be constructed for the prediction of the test dataset. Random forest runs efficiently in large dataset. It can handle thousands of input variables without variable deletion. It also handles the missing values inside the dataset for training the model. It is difficult to handle the unbalanced dataset by using Random Forest.

REQUIREMENT ANALYSIS:-

(i) FUNCTIONAL REQUREMENTS

In software engineering, a functional requirement defines a function of the software system or its components. A function is described as a set of inputs, the behavior and outputs. Functional requirements maybe calculations, technical details, data manipulation and processing and other specific functionality that define what a system must accomplish. A functional requirement defines a function of a software system or its components. It

captures the intended behavior of the system. This behavior maybe expressed in terms of services, tasks or functions that the system has to perform.

(ii) Non Functional Requirements

Non-Functional Requirements specify the criteria that can be used to judge the operation of a system, rather than specific behaviors. They are the metrics that are considered to measure the performance of the developed system. The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates.

DESIGN

DESIGN GOALS

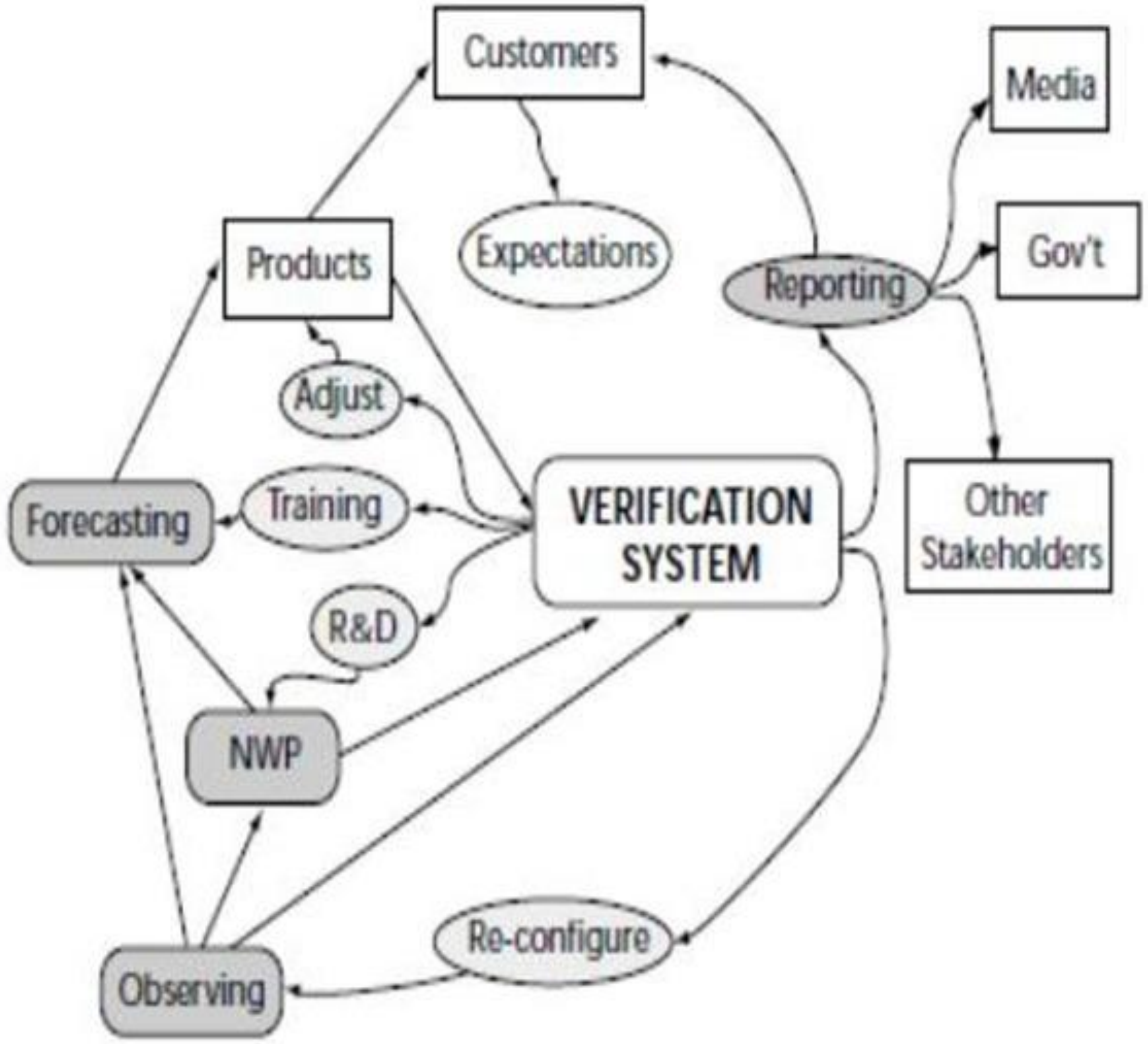
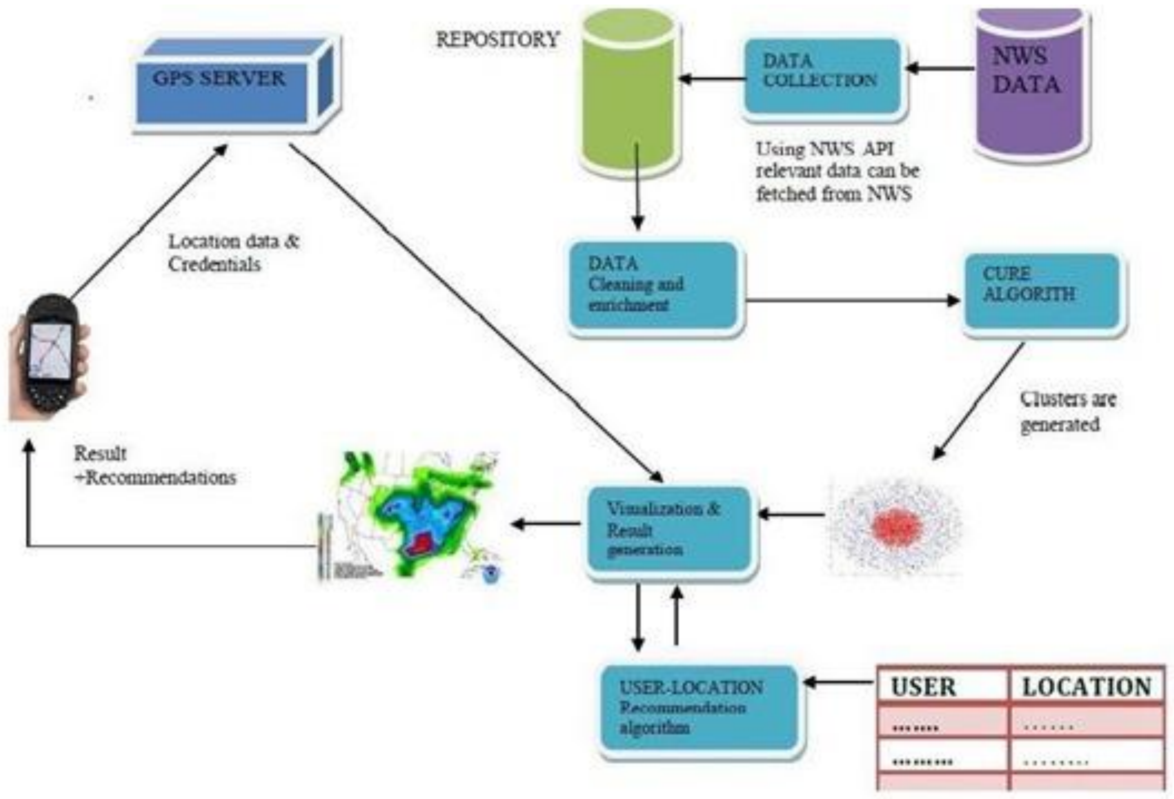
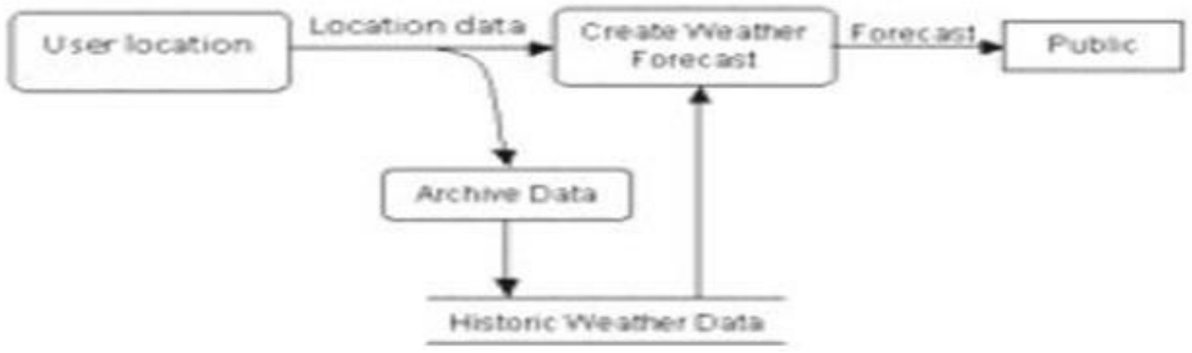


Fig no:- Design Goals

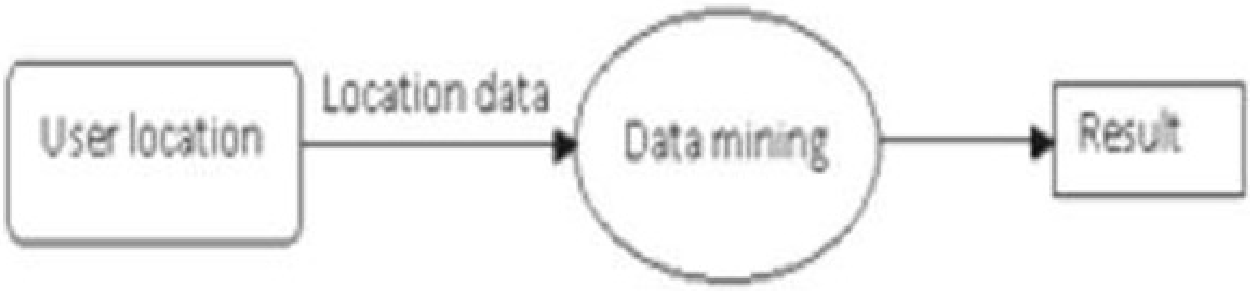

GPS SERVER
REPOSITORY
DATA COLLECTION
NWS DATA
Using NWS API relevant data can be fetched from NWS
Location data & Credentials
DATA Cleaning and enrichment
CURE ALGORITH
Clusters are generated
Result +Recommendations
Visualization & Result generation
USER-LOCATION Recommendation algorithm
USER
LOCATION

OVERALL SYSTEM ARCHITECTURE:-

Fig no: Overall System Architecture



DATA FLOW DIAGRAM/ACTIVITY DIAGRAM



z

Level 0

Fig no: Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the flow of data through an information system. A data flow diagram can also be used for the visualization of data processing (structured design). It is common practice for a designer to draw a context-level DFD first which shows the interaction between the system and outside entities. This context-level DFD is then exploded to show more detail of the system being modelled.

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system. The DFD is simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data and the output data generated by the system.

A DFD model uses a very limited number of primitive symbols to represent the functions performed by a system and the data flow among the functions. The main reason why the DFD technique is so popular is probably because of the fact that DFD is a very simple formalism- It is simple to understand and use. Starting with the set of high level functions that a system performs, a DFD model hierarchically represents various sub functions. In fact, any hierarchical model is simple to understand. The human mind is such that it can easily understand any hierarchical model of a system because in a hierarchical model, starting with a very simple and abstract model of system, different details of a system are slowly introduced through the different hierarchies. The data flow diagramming technique also follows a simple set of intuitive concepts and rules.

DFD is an elegant modelling technique that turns out to be useful not only to represent the results of a structured analysis of a software problem but also for several other applications such as showing the flow of documents or items in an organization. A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for

the visualization of data processing (structured design). On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.

Level 0: A context-level or level 0 data flow diagram shows the interaction between the system and external agents which act as data sources and data sinks. On the context diagram (also known as the Level 0 DFD) the system's interactions with the outside world are modelled purely in terms of data flows across the system boundary. The context diagram shows the entire system as a single process, and gives no clues as to its internal organization

Level 1: The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, Level 1 Data Flow diagram shows an in-depth explanation of overall process of the data flow.

IMPLEMENTATION

FUNCTIONALITY

We can divide our process in two modules namely:

1) Weather Mining

2) Recommendation

B. Weather Mining:

Data collection: We have collected weather data from WORLD DATA CENTER for climate, Hamburg. We have decided to use NWS API for data collection in future.

Data formatting and cleaning: We have converted our data from .NC (netcdf) format to .CSV (comma-separated values) format because WEKA supports .CSV format.

Clustering: Using WEKA, we have performed clustering on weather data to draw inferences.

Recommendation: We have planned to use recommendation algorithm as user to location collaborative algorithm similar to user to item collaborative algorithm. This algorithm uses user location (N*M) metrics.

Visualization: To generate visualization for user, we have used NOAA weather and climate tool kit.

For the part of the implementation, on which your project focused most, which algorithms you implemented or used and if any modifications were needed to those algorithms or if you did some initial pre-processing, discuss

here For the other phases of data mining, discuss brief. E.g., if you focused most on visualization, you can talk about: which data (Example: downloaded from some website put the URL here; did some survey, then talk about how you did the survey etc) collection approach was used in the project?

C. Recommendation

Extract the location of the user. Extract the destination of the user and then recommend the best path according to the conditions.

CONCLUSION

Traditionally, weather forecasting has always been performed by physically simulating the atmosphere as a fluid and then the current state of the atmosphere would be sampled. In the previous system the future state of the atmosphere is computed by solving numerical equations of thermodynamics. But this model is sometimes unstable under disturbances and uncertainties while measuring the initial conditions of the atmosphere. This leads to an incomplete understanding of the atmospheric processes, so it restricts weather prediction.

Our proposed solution of using Machine learning for weather predicting is relatively robust to most atmospheric disturbances when compared to traditional methods. Another advantage of using machine learning is that it is not dependent on the physical laws of atmospheric processes. In the long run weather prediction using Machine Learning has a lot of advantages and thus it should be used globally.

FURTHER ENHANCEMENTS

Weather forecasts are becoming more detailed, more accurate and are providing the information needed to make sound decisions to protect life and property. Technological advances, such as apps, are making weather information more accessible and immediately alerting those in harm’s way.

The current version of Weather Prediction that we have developed is still premature. This implies that there are still many limitations that can be resolved and improved. One of the biggest limitation right now is, that the location has to be chosen from the list of places the application is bound to. This can be improved if we use web scraping tools to automatically get the weather data, for various locations, from the internet and then input it into the database. Another enhancement that can be done is the automatic validation of longitude and latitude coordinates. Another improvement that can be done is to beautify the UI to make it more appealing to the younger generation.

Future enhancements will make our Weather Prediction more flexible, user friendly and thus it will be more appealing to a wide range of audience.

Machine Learning Techniques In this research, as the predicted outcomes are continuous numeric values, temperature in our case, we use regression technique. We find that Random Forest Regression (RFR) is the superior regressor, as it ensembles multiple decision trees while making decision. In addition, we show comparison of several other state-of-the-art ML techniques with the RFR technique. The incorporated regression techniques are Ridge Regression (Ridge), Support Vector (SVR), Multi-layer Perceptron (MLPR), and Extra-Tree Regression (ETR).

Data Pre-processing

After having the raw data from 'underground', we make sure that each row (record) in the dataset has records for all ten cities for a particular timestamp. We eliminate any feature with empty or invalid data while creating the dataset. Also, we convert the categorical features in the dataset, such as wind direction and condition, into dummy/indicator variables using a technique called 'One Hot Encoding' [3]. We perform this conversion prior to the separation of training and test data. This is because, in both training and test data, we need the same number of feature variables. If we do this conversion after the separation, then there remains no guarantee that both of them will have all the categorical values for the corresponding features. If the number of categorical values for training and test sets is not the same, then the conversion yields to a different number of features for these sets. That is why we need to perform this conversion before the separation of training and test datasets. Furthermore, we perform mean scaling $x \leftarrow x-\mu \ \sigma$ to all the continuous variables so that the variables possess approximately zero mean, which in practice, reduces computational cost while training the module.

2.5. DESIGN APPROACH

Model-View-Controller

MVC is an application design model comprised of three interconnected parts. They include the Model (data), View (user interface).Controller (processes that handle input). -Commonly used for developing the modern user interface.

-It works well with object-oriented programming since the different models, views, and controllers can be treated as objects and reused within an application.

Advantages Of MVC

- **Faster Web Application Development Process: .**
- **MVC Web Application Supports Asynchronous Technique**
- **Offers The Multiple Views:**
- **Ideal for developing large size web application: ...**
- **MVC Model Returns The Data Without The Need of Formatting.**

Disadvantage of MVC

- **Difficulty of using MVC with modern user interface**
- **Knowledge on multiple technologies is required.**
- **Developer have knowledge of client side code and html code**
- **Need multiple programmers.**

RISKS :

Design Constraints, Assumptions and Dependencies

The working of the application has a limitation to take inputs as few previous days data with the current day and predict the next days climate.

User constraints : Should have to be a registered user. Access to redirect and allows to use the location that is available in out server only. Internet connection to use the application.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FUNCTIONAL REQUREMENTS

In software engineering, a functional requirement defines a function of the software system or its components. A function is described as a set of inputs, the behavior and outputs. Functional requirements maybe calculations, technical details, data manipulation and processing and other specific functionality that define what a system must accomplish. A functional requirement defines a function of a software system or its components. It captures the intended behavior of the system. This behavior maybe expressed in terms of services, tasks or functions that the system has to perform.

3.2 Non Functional Requirements

Non Functional Requirements specify the criteria that can be used to judge the operation of a system, rather than specific behaviors. They are the metrics that are considered to measure the performance of the developed system. The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates.

Three key considerations involved in the feasibility analysis are

- **Usability: Simple is the key here. The system must be simple that people like to use it, but not so complex that people avoid using it. The user must be familiar with the user interfaces and should not have problems in migrating to a new system with a new environment. The menus, buttons and dialog boxes should be named in a manner that they provide clear understanding of the functionality.**
- **Reliability: The system should be trustworthy and reliable in providing the functionalities.**

- **Performance:** The system is going to be used by many people simultaneously. Performance becomes a major concern. The system should not succumb when many users would be using it simultaneously. It should allow fast accessibility to all of its users.
- **Scalability:** The system should be scalable enough to add new functionalities at a later stage. There should be a common channel, which can accommodate the new functionalities.
- **Maintainability:** The system monitoring and maintenance should be simple and objective in its approach.
- **Portability:** The system should be easily portable to another system.
- **Reusability:** The system should be divided into such modules that it could be used as a part of another system without requiring much of work.
- **Security**: Security is a major concern .This system must not allow unauthorized users to access the information of other users.

3.3 DOMAIN AND UIREQUIREMENTS

- **Admin Login: -** Admin will login to the system using Admin ID and Password.
- **Add previous weather data: -** Admin will add previous weather data in database.
- **User Login: -** User can access the system using his user ID and password.
- **User Registration: -** User can register himself by filling registration form.

- **Input Parameters: - In this module user will enter parameters such as temperature, humidity and wind.**
- **Weather Forecast: - In this module, system will take current parameters entered by the user and will compare the parameters with the data in database and will predict the weather.**

3.4 HARDWARE AND SOFTWARE REQUIREMENTS

3.4. HARDWARE REQUIREMENTS:

The hardware requirements listed below are almost in a significantly higher level which represents the ideal situations to run the system. Following are the system hardware requirements used :

Processor	**- Pentium –III**
Speed	**- 1.1 GHz**
RAM	**- 256 MB(min)**
Hard Disk	**- 20 GB**
Floppy Drive	**- 1.44 MB**
Key Board	**- Standard Windows Keyboard**
Mouse	**- Two or Three Button Mouse**
Monitor	**- SVGA**

3.5. SOFTWARE REQUIREMENTS:

A major element in building a system is a section of compatible software since the software in the market is experiencing in geometric progression. Selected software should be acceptable by the firm and the user as well as it should be feasible for the system. This document gives the detailed description of the software requirements specification. The study of requirement specification is focused specially on the functioning of the system. It allows the analyst to understand the system, functions to be carried out and the performance level which has to be maintained including the interfaces established.

Operating System	**- Windows**
Application Server	**- Tomcat5.0/6.X**
Front End	**- HTML, Java, Jsp**
Scripts	**- JavaScript.**
Server side Script	**- Java Server Pages.**
Database	**- Mysql**
Database Connectivity	**- JDBC.**

3.6. DATA REQUIREMENTS:

Data collection: We have collected weather data from WORLD DATA CENTER for climate, Hamburg. We have decided to use NWS API for data collection in future.

Data formatting and cleaning: We have converted our data from .NC (netcdf) format to .CSV (comma-separated values) format because WEKA supports .CSV formats.

4.1 STATE MACHINE UMLDIAGRAM

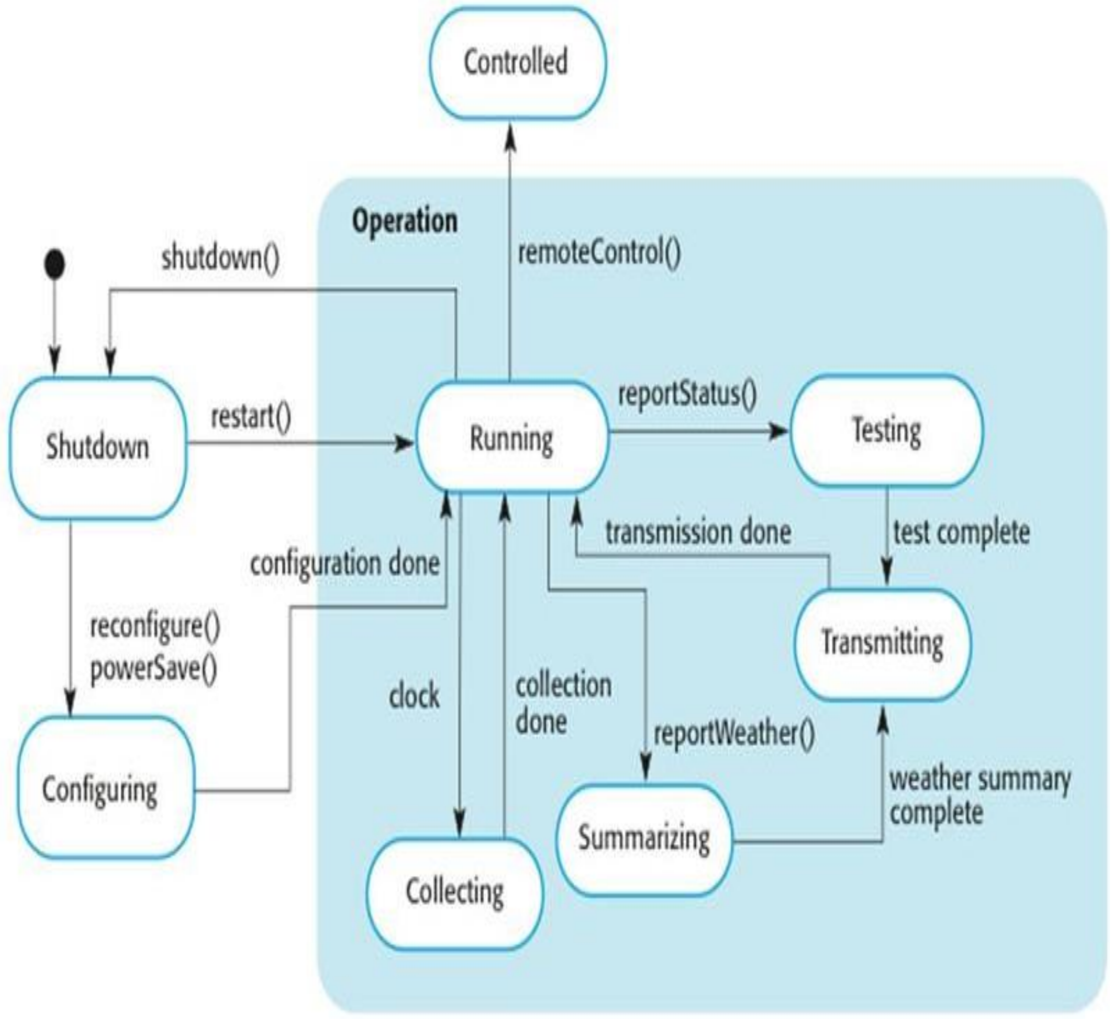


Fig no.4.4. State Machine UML Diagram

4.2 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Chart. Sequence diagrams are sometimes called event diagrams, event scenarios and the sequence diagram for this project is given in figure

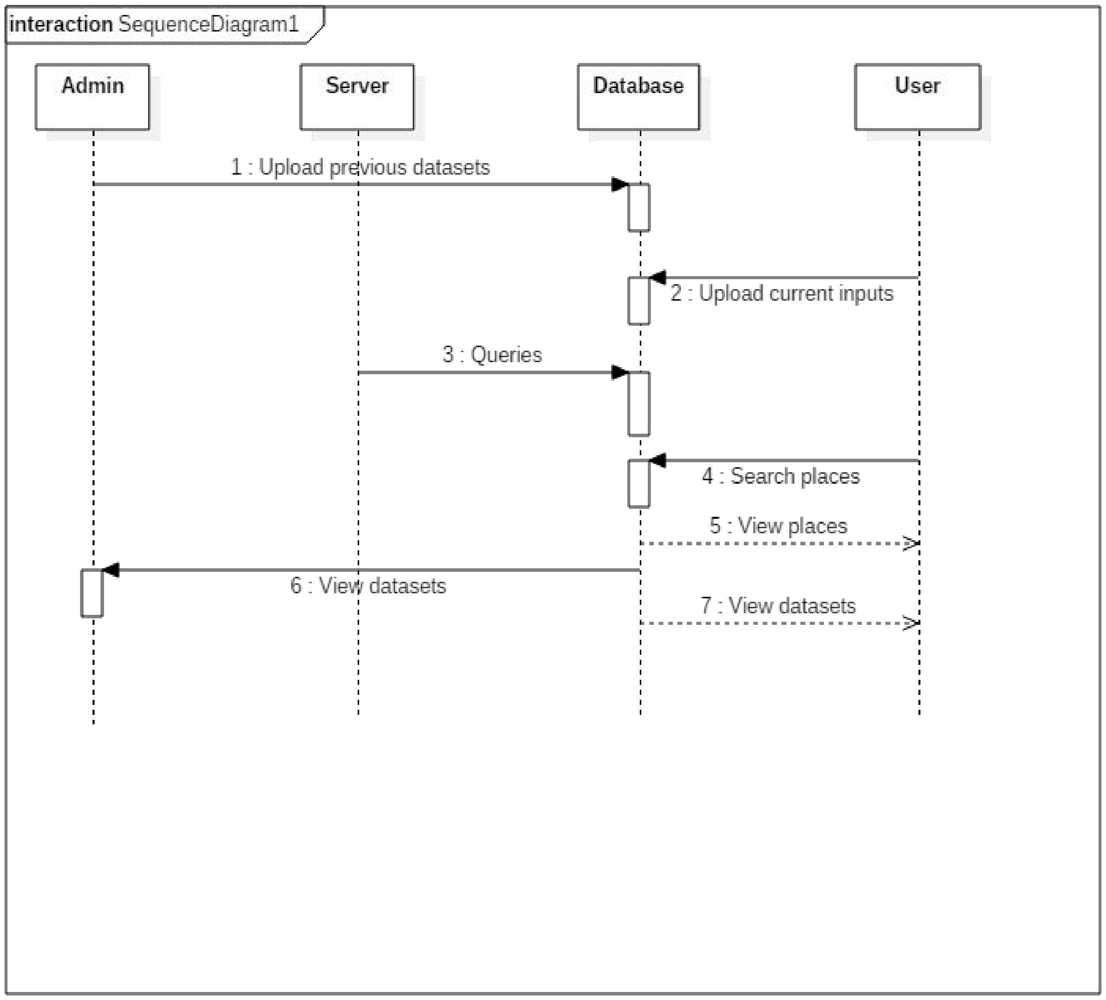


Fig no .4.5. Sequence Diagram

4.3 INTERACTION OVERVIEW DIAGRAM

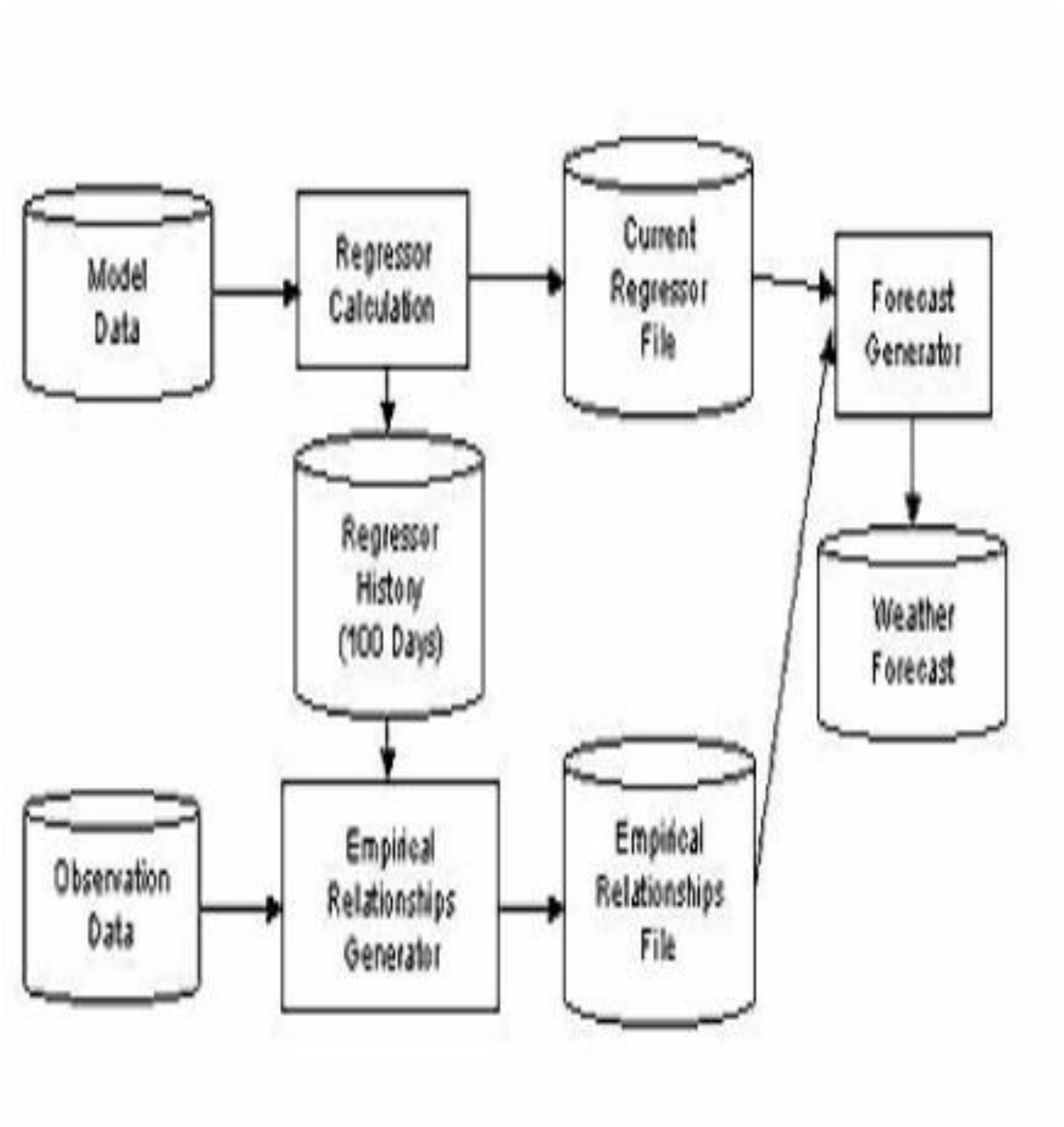
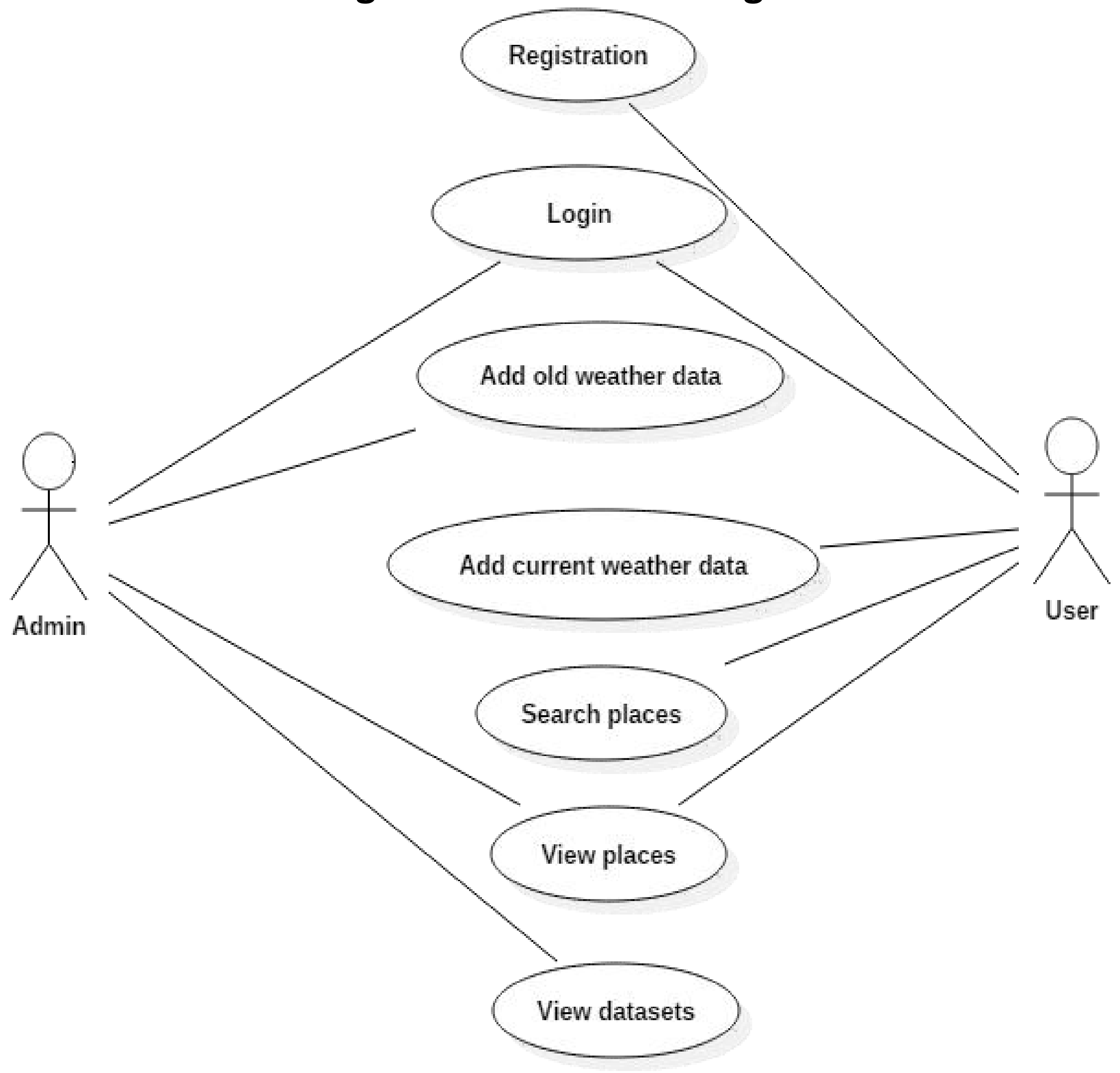


Fig no. 4.6. Interaction Overview Diagram

4.4 USE CASE DIAGRAM

A use case diagram is a type of behavioral diagram created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. Use case diagram provides us the information about how that users and use cases are related to the system.

Fig no.4.7. Use Case Diagram



Data collection and processing

```
def extract_weather_data(url, api_key, target_date, days):
    records = []
    for _ in range(days):
        request = BASE_URL.format(API_KEY, target_date.strftime('%Y%m%d'))
        response = requests.get(request)
        if response.status_code == 200:
            data = response.json()['history']['dailysummary'][0]
            records.append(DailySummary(
                date=target_date,
                meantempm=data['meantempm'],
                meandewptm=data['meandewptm'],
                meanpressurem=data['meanpressurem'],
                maxhumidity=data['maxhumidity'],
                minhumidity=data['minhumidity'],
                maxtempm=data['maxtempm'],
                mintempm=data['mintempm'],
                maxdewptm=data['maxdewptm'],
                mindewptm=data['mindewptm'],
                maxpressurem=data['maxpressurem'],
                minpressurem=data['minpressurem'],
                precipm=data['precipm']))
            time.sleep(6)
            target_date += timedelta(days=1)
        return records
    if not station_id:
                raise Exception("'station_id' is required.")
            if 'recordId' in params and 'current' in params:
                raise Exception("Cannot have both 'current' and 'recordId'")
            if 'start' in params:
                start = params['start']
                self.parse_param_timestamp(start)
                if len(start) < 19:
                    start = '{}T00:00:00Z'.format(start[:10])
                elif len(params['start']) < 20:
                    start = start.replace(' ', 'T')
                    start = '{}Z'.format(start)
                params['start'] = start
            if 'end' in params:
                end = params['end']
                self.parse_param_timestamp(end)
                if len(end) < 19:
                    end = '{}T23:59:59Z'.format(end[:10])
                elif len(params['end']) < 20:
```

```
        end = end.replace(' ', 'T')

        end = '{}Z'.format(end)

    params['end'] = end


request_uri = "/stations/{stationId}/observations".format(

    stationId=station_id)


if len(params) > 0:

    if 'recordId' in params:

        return self.make_get_request(

            '{old_request_uri}/{recordId}'.format(

                old_request_uri=request_uri,

                recordId=params['recordId']),

            end_point=self.DEFAULT_END_POINT)

    if 'current' in params:

        return self.make_get_request(

            '{old_request_uri}/current'.format(

                old_request_uri=request_uri),

            end_point=self.DEFAULT_END_POINT)


    if len(params) > 1:

        request_uri = '{old_request_uri}?{query_string}'.format(

            old_request_uri=request_uri,
                query_string=urlencode(params))


            observations = self.make_get_request(
                request_uri, end_point=self.DEFAULT_END_POINT)
            if 'features' not in observations:
                raise Exception(observations)
            return observations['features']


        return self.make_get_request(
"/stations/{stationId}/observations".format(stationId=station_id),
            end_point=self.DEFAULT_END_POINT)


    def products(self, id):
        functional tests @todo(paulokuong) later on.
  Args:
            id (str): product id.
        Returns:
            json: json response from api.
"""
        return self.make_get_request(
"/products/{productId}".format(productId=id),
            end_point=self.DEFAULT_END_POINT)
  def products_types(self, **params),
   Response in this method should not be modified.
     In this way, we can keep track of changes made by NOAA through
     functional tests @todo(paulokuong) later on.
Args:
       type_id (str): an id of a valid product type
       locations (boolean[optional]): True to get a list of
          locations that have issues products for a type.
       location_id (str): location id.
     Returns:
       json: json response from api.
     if 'type_id' in params and 'locations' not in params:
       return self.make_get_request(
"/products/types/{type_id}".format(type_id=params['type_id']),
          end_point=self.DEFAULT_END_POINT)
     elif 'locations' in params:
       if 'type_id' not in params:
          raise Exception('Error: Missing type id (type_id=None)')
       if 'location_id' in params:
          return self.make_get_request(
            ('/products/types/{type_id}/locations/'
            '{location_id}').format(
               type_id=params['type_id'],
                    location_id=params['location_id']),
                end_point=self.DEFAULT_END_POINT)
        else:
            return self.make_get_request(
"/products/types/{type_id}/locations".format(
                    type_id=params['type_id']),
                end_point=self.DEFAULT_END_POINT)
      return self.make_get_request(
"/products/types",
        end_point=self.DEFAULT_END_POINT)
 def products_locations(self, **params)
      functional tests @todo(paulokuong) later on.
Args:
        location_id (str): location id.
      Returns:
        json: json response from api.
      if 'location_id' in params:
        return self.make_get_request(
"/products/locations/{locationId}/types".format(
                locationId=params['location_id']),
            end_point=self.DEFAULT_END_POINT)
      return self.make_get_request(
"/products/locations",
            end_point=self.DEFAULT_END_POINT)
  def offices(self, office_id):
        Response in this method should not be modified.
        In this way, we can keep track of changes made by NOAA through
        functional tests @todo(paulokuong) later on.
 Args:
            office_id (str): office id.
        Returns:
            json: json response from api.
        return self.make_get_request("/offices/{office_id}".format(
            office_id=office_id), end_point=self.DEFAULT_END_POINT)
  def zones(self, type, zone_id, forecast=False):
        functional tests @todo(paulokuong) later on.
  Args:
            type (str): a valid zone type (list forthcoming).
            zone_id (str): a zone id (list forthcoming).
            forecast (bool): True to show forecast data of the zone.
        Returns:
            json: json response from api.
        if forecast:
            return self.make_get_request(
"/zones/{type}/{zone_id}/forecast".format(
                type=type, zone_id=zone_id),
            end_point=self.DEFAULT_END_POINT)
        return self.make_get_request("/zones/{type}/{zone_id}".format(
            type=type, zone_id=zone_id), end_point=self.DEFAULT_END_POINT)
    def alerts(self, **params):
        Response in this method should not be modified.
        In this way, we can keep track of changes made by NOAA through
        functional tests @todo(paulokuong) later on.
Args:
            alert_id (str): alert id.
            active (int): Active alerts (1 or 0).
            start (str): Start time (ISO8601DateTime).
            end (str): End time (ISO8601DateTime).
            status (str): Event status (alert, update, cancel).
            type (str): Event type (list forthcoming).
            zone_type (str): Zone type (land or marine).
            point (str): Point (latitude,longitude).
            region (str): Region code (list forthcoming).
            state (str): State/marine code (list forthcoming).
            zone (str): Zone Id (forecast or county, list forthcoming).
            urgency (str): Urgency (expected, immediate).
            severity (str): Severity (minor, moderate, severe).
            certainty (str): Certainty (likely, observed).
            limit (int) Limit (an integer).
        Returns:
            json: json response from api.
       if 'alert_id' in params:
            return self.make_get_request(
"/alerts/{alert_id}".format(alert_id=params['alert_id']),
                end_point=self.DEFAULT_END_POINT)
        return self.make_get_request(
"/alerts?{query_string}".format(query_string=urlencode(params)),
            end_point=self.DEFAULT_END_POINT)
  def active_alerts(self, count=False, **params):
        functional tests @todo(paulokuong) later on.
   Args:
            count (bool): True to hit /alerts/active/count.
            zone_id (str): a valid zone, see list in counts endpoint.
            area (str): a valid area, see list in counts endpoint.
            region (str): a valid region, see list in counts endpoint
        Returns:
            json: json response from api.
        if count:
            return self.make_get_request(
"/alerts/count",
                end_point=self.DEFAULT_END_POINT)
        if 'zone_id' in params:
            return self.make_get_request(
    "/alerts/active/zone/{zoneId}".format(
                zoneId=params['zone_id']),
              end_point=self.DEFAULT_END_POINT)
        if 'area' in params:
            return self.make_get_request(
    "/alerts/active/area/{area}".format(
                area=params['area']),
              end_point=self.DEFAULT_END_POINT)
        if 'region' in params:
            return self.make_get_request(
    "/alerts/active/region/{region}".format(
                region=params['region']),
              end_point=self.DEFAULT_END_POINT
```

REFERENCES

1. A. Payne and P. Frow, “A strategic framework for customer relationship management,” J. Marketing, vol. 69, no. 4, pp. 167–176, Oct. 2005.

2. L. D. Xu, “Enterprise systems: State-of-the-art and future trends,” IEEE Trans. Ind. Inf., vol. 7, no. 4, pp. 630–640, Nov. 2011.

3. A. Berson, K. Thearling, and S. Smith, Building Data Mining Applications for CRM. New York: McGraw-Hill, 1999.

4. K. Coussement and D. V. Poel, “Churn prediction in subscription services: An application of support vector machines while comparing two parameter-selection techniques,” Expert Syst. Appl., vol. 34, no. 1, pp. 313–327, Jan. 2008.

5. W. Verbeke, K. Dejaeger, D. Martens, J. Hur, and B. Baesens, “New insights into churn prediction in the telecommunication sector: A profit driven data mining approach,” Eur. J. Oper. Res., vol. 218, no. 1, pp. 211–229, Apr. 2012.

6. W. J. Reinartz and V. Kumar, “The impact of customer relationship characteristics on profitable lifetime duration,” J. Marketing, vol. 67, no. 1, pp. 77–99, Jan. 2003.

7. P. Datta, B. Masand, D. R. Mani, and B. Li, “Automated cellular modeling and prediction on a large scale,” Artif. Intell. Rev., vol. 14, no. 6, pp. 485–502, Dec. 2000.

8. D. Popović and B.D. Bašić, “Churn prediction model in retail banking using fuzzy C-means algorithm,” Informatica, vol. 33, no. 2, pp. 235–239, May 2009.

9. C.-P. Wei and I. T. Chiu, “Turning telecommunications call details to churn prediction: A data mining approach,” Expert Syst. Appl., vol. 23, no. 2, pp. 103–112, Aug. 2002.

10. M. Owczarczuk, “Churn models for prepaid customers in the cellular telecommunication industry using large datamarts,” Expert Syst. Appl., vol. 37, no. 6, pp. 4710–4712, Jun. 2010.

11. Weather Forecasting Using Deep Learning Techniques - IEEE Conference Publication, ieeexplore.ieee.org/document/7415154.

12. “Weather.” Kaggle, www.kaggle.com/tags/weather.

13. “Weather Forecasting.” Kaggle, www.kaggle.com/questions-and-answers/27537.