


SmartSurfPredictor: Report

6 November 2015

Eeswari Avudainathan (12118055)


Marcel Bühler (99211688)

Smart Surf Predictor



Welcome to Smart Surf Predictor

Forecast Feedback

Smart Surf Predictor

Forecast Date: 23/10/2015

Select forecast spot:

Newcastle

The Farm

Sydney (Cronulla)

Sydney (Bondi)

Puerto Rico (Cuba)

Sydney (Manly)

Sydney (Maroubra)

Back

Surf Spot: The Farm

Minumum Height (ft): 4

Maximum Height (ft): 7

Wave Ratings: ★ ★

Swell Height (ft): 7.5

Swell Rate (seconds): 8

Swell Direction: SE

Wind Speed (mph): 18

Wind Direction: S

Weather: ☁

Temperature (celsius): 16.0

Surfing condition are: ●

Do you like this prediction ?

Yes

No

Acknowledgement: Data kindly provided by Magicseaweed.

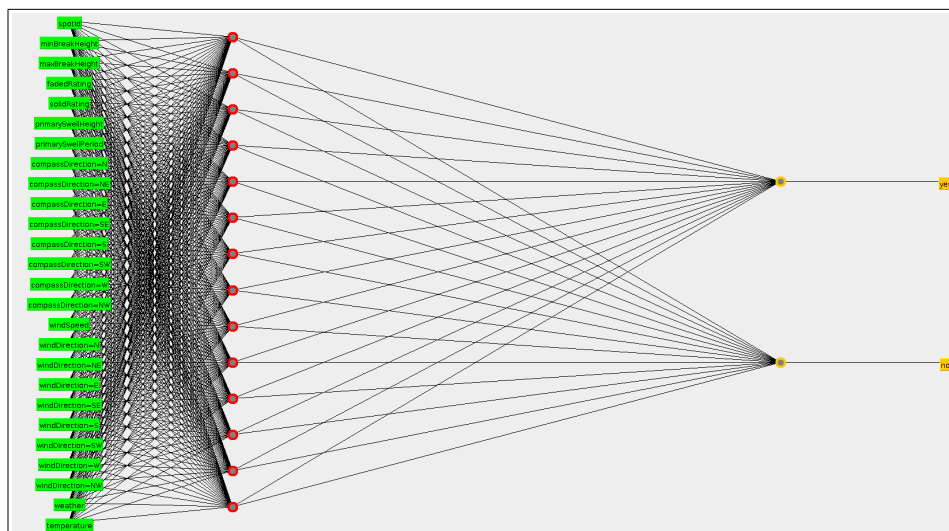


Table of Contents

1. Context.....	3
1.1 Choice of topic.....	3
1.2 The Big Picture and its relationship to the Demonstrator.....	3
2. Design.....	5
2.1 Relation to existing systems.....	5
2.2 Development.....	5
1. Collecting ideas.....	5
2. Research.....	5
3. Planning / Requirements engineering.....	6
4. Milestone 1: Basic implementation.....	7
5. Milestone 2: Optimization.....	7
6. Milestone 3: Improving user friendliness and code clean up.....	8
7. Coding intensity.....	9
2.3 Team spirit and team organization.....	9
2.4 Tools.....	10
2.5 Improving intelligence and feedback learning.....	10
1. Improving sample data set.....	10
2. Classifier evaluation.....	10
3. Feedback learning.....	12
3. Operation Manual.....	13
3.1 General information.....	13
3.2 How to install SmartSurfPredictor on your computer.....	13
3.3 Welcome to SmartSurfPredictor: main view.....	14
3.4 Set-up Wizard for SmartSurfPredictor.....	15
3.5 Second feature on main view : “Add/Change Surf Locations”.....	18
3.6 Second feature on main view : “Rate more Forecast”.....	19
3.7 The last feature on main view: “Show Forecast”.....	20
3.8 Error Checking Mechanism for “Show Forecast” feature.....	21
3.9 Console Output.....	22
4. Appendix.....	23
4.1 Class diagram: Forecast data modelling.....	23
4.2 Overview of file names and locations.....	24
1. System files.....	24
2. User files.....	24
3. Test files.....	24
4.3 Attributes used for classifier.....	25
4.4 Sequence diagram.....	26
5. References / Links.....	27

1. Context

1.1 Choice of topic

We as surfers browse multiple websites in order to find swell forecast information. This takes a lot of time. We intended to build an agent that would intelligently find this information for us and other surfers and analyse them based on surfing preferences and skills.

1.2 The Big Picture and its relationship to the Demonstrator

The Big Picture is meant to be a mobile application (iOS, Android), as well as a webapp running on the magicseaweed website (<http://magicseaweed.com/>). The SmartSurfPredictor Demonstrator runs as a local Java application. It does provide the basic functionality: It is able to learn from user feedbacks, pro-actively fetch forecasts, label them and display them. For modelling the user preferences, the Demonstrator's Neural Network uses 13 features, such as swell height, temperature, swell rate, swell direction, wind conditions, etc. A detailed overview and description about the used attributes can be found in the appendix.

The Big Picture would be a project running in cooperation with magicseaweed (msw). Many users already have a msw account and this would give SmartSurfPredictor much more information. With this information, the service could be improved and more features could be taken into consideration when creating individual forecast ratings. It would also be possible to recommend forecasts not only for the next day, but a for the next week. Another aspect where the Big Picture differs from the Demonstrator is the number of surf spots. In the Demonstrator, it is limited for licensing reasons. The Big Picture would offer recommendations for all surf spots on msw.

In terms of performance, the learning/recommender algorithm has space for improvement. Our thoughts about this is described in more detail in the SmartSurfPredictor critique below. For more details about the current learning algorithm, please read *Improving intelligence and feedback learning* on page 10.

The SmartSurfPredictor Big Picture would implement many nice and handy features, such as SmartSpotPredictor (recommend new surf locations), SmartEquipmentPredictor (recommend the appropriate surf equipment for each weather and surf), SmartBuddyPredictor (create a surfing community near you by connecting you with other SmartSurfPredictor users) or SmartTransportPredictor (suggest the best way to get to the surf spots).

The Demonstrator focuses on providing the basic functionality in order to give a sample of test users and the company msw a feel of what the Big Picture could be like. With the Demonstrator we are mainly targeting surf companies (especially msw) and our future customers (surfers of any skill level).

We chose to implement this portion as Demonstrator because it implements the core idea of SmartSurfPredictor (a personalized recommender system for surf forecasts). Our aim was to create a software that we were actually able to use in our daily surf life.

What the users can expect from the Big Picture is the following: The Big Picture provides accurate and personalized surf forecast recommendations. It is also very easy to get started. After downloading the app / starting the webapp, a set-up wizard guides the user through the initial steps. The aim is to provide the user with an accurate feedback only a few minutes after registration.

2. Design

2.1 Relation to existing systems

The SmartSurfPredictor Demonstrator was built as a stand-alone agent. It does not depend on other agents. It uses the msw API in order to retrieve forecast data. The Big Picture could be integrated into the magicseaweed website and its already existing mobile application. magicseaweed operates a website where users can open an account. SmartSurfPredictor could be integrated as a nice feature for the registered users and offer them personalized surf forecasts. Neither the msw website nor the msw mobile app offer a personalized surf forecast recommender yet. Integrating SmartSurfPredictor would therefore certainly enhance the user experience.

2.2 Development

1. *Collecting ideas*

From scratch, our aim was creating an intelligent agent that would be able to add value to something already existing. So we had to find a project idea that met the criteria of being an intelligent agent and that was actually going to be useful. The idea of a notification agent for surf forecasts came up. A rule-based notification agent itself, however, would have been very basic and from our point of view not intelligent. So we decided to create an intelligent and personalized agent that is able to deliver individually selected surf forecasts. This would also offer us the possibility to investigate machine learning techniques and see them work in a real life application.

2. *Research*

After finding out about our main idea, it was time to write the proposal. One important challenge we faced was finding forecast data. Ideally we would have access to real forecasts without the need of crawling the web and parsing websites already containing surf forecasts. We searched the internet and found websites that offer access to their API:

- <http://www.worldweatheronline.com/api/marine-weather-api.aspx>
- <http://swellcast.com.au/surf-forecast-api>
- <http://magicseaweed.com/developer/forecast-api>

magicseaweed is a wide-spread and very popular company and their API seemed to be the most useful and best documented one. We requested non-commercial access to the API by sending an e-mail. Hereupon we received an answer from Eoghan O'Loughlin containing an API key.

magicseaweed runs a RESTful service that returns data in the JSON format. The response always contains all the forecasts for the next couple of days (in a three-hour interval). So each request returns a considerable amount of data. As we were going to use this data in our Java application so we had to figure out a way to efficiently convert it from JSON to Java objects. After some research we decided to use the google http-client-api with the jackson 2 extension. This would allow us to automatically parse the JSON response using “@Key” annotations and directly create Java objects from it using the JacksonFactory.

The third aspect was finding an appropriate way to rate surf forecasts according to user preferences. We wanted to use a pre-existing library that allowed us to apply common machine learning techniques. After some research we discussed two alternatives: the Weka library and the Java Machine Learning Library (Java-ML). We decided to use Weka because it seemed easier to use and included a “explorer”-mode that allowed testing various algorithms and visualizing the results without the need to write additional Java code.

3. Planning / Requirements engineering

Developing a full-fledged application requires a significant amount of planning. We started by writing user stories, designing the work flow and identifying the components needed. In order to keep an overview of the work flow, we designed a UML sequence diagram illustrating the sequence of method calls. The diagram can be found in the appendix on page 26.

We decided to use the MVC (Model-View-Controller) software design pattern because it was perfectly suitable for our application. The “model” package would contain all the data classes and the business logic. The “view” package would deal with the graphical user interface. The “controller” package would include the actionListeners for the GUI and the REST controller, that makes requests to the msw API. We also would use a package (“util”) containing static constants and methods and a “test” package that would be used to independently run only parts of the application and testing their behaviour using J Unit tests.

We divided the implementation into the following 5 main tasks:

#	Task	Description	Assignee
1	API requests and data conversion	Fetch live forecast data from msw and convert it into Java objects	Marcel
2	Classifier selection and training	Data preprocessing, integrating the Weka library, classifier performance analysis	Marcel
3	Notification agent	Implementation of automatic notifications twice a day	Eeswari
4	Graphical User Interface	GUI with five elements: <ul style="list-style-type: none"> • Welcome screen • Spot selection • Feedback rating • Pop-up • Set-Up wizard 	Eeswari
5	Writers and Readers	Read and write user data as well as application resource files.	Eeswari / Marcel

Table 1: Implementation: Main implementation tasks

The implementation was divided into three milestones, which are described in detail below.

4. Milestone 1: Basic implementation

The basic implementation should be capable of making API requests to magicseaweed (msw) and retrieve the data in JSON format (task #1). The data should be converted into Java instances (see appendix for UML class diagram describing forecast data on page 23). Furthermore, it should be possible to rate forecasts in a simple UI (task #4) and write the labelled data (training set) to a folder in the local file system (task #5).

As the next step, the classifier should be loading the data from the file and rate new live forecasts from msw (task #2). Furthermore, a simple notification thread (task #3) should regularly check the time and launch the notification process twice a day (at 9am and 3pm).

We started early and could finish all the mentioned steps above in time.

5. Milestone 2: Optimization

Milestone 2 was about improving the efficiency and robustness of the code. In the basic implementation from milestone 1, the data used for the initial rating was fetched as random forecasts from msw. This had two major disadvantages:

- The time it took to retrieve 20 forecasts (found to be 3-4 seconds).

- The similarity of the forecasts. Random forecasts did not usually represent a broad range of values for each feature. The classifier, trained with 20 often similar forecasts, did not generalize well most of the time.

For these reasons we decided to create a more balanced dataset for receiving the initial user feedback where a broad range of values for each feature could be covered. This solved both problems at the same time: loading the data from a local file was much faster than retrieving it from the online API and the classifier, trained with the new data, generalized much better. Additionally, we created four different datasets and tested them with various classifiers. The aim was to find the classifier that performed best with a small dataset between 20 and 90 training examples. The classifier evaluation is described in more detail below.

Another big part of Milestone 2 was increasing the robustness of the processes (exception handling and edge cases). The data retrieved from msw was not always complete. This sometimes caused errors and the application crashed when converting the data into Java instances. A similar situation occurred when training the classifier without creating the training set beforehand or not having specified favourite spots. These problems were solved through exception handling and edge case testing.

Furthermore the rest of the GUI was implemented including a welcome screen, the spot selection window and the notification pop-up.

6. Milestone 3: Improving user friendliness and code clean up

One point that had to be optimized in Milestone 3 was the dealing with file names. In the planning phase, we did not specifically discuss file names and paths. In the development process, this sometimes caused confusion. An actual overview of the file names and locations can be found in the appendix (page 24).

A big part of Milestone 3 was improving the user friendliness of the SmartSurfPredictor application. The GUI was improved (adding pictures, exit button). We also added Exception dialogues in case the user wants to retrieve and rate forecasts without having set up the application properly (selecting favourite spots and rating at least 20 forecasts).

Before exporting the application, unnecessary libraries and docs were removed, reducing the exported file size from more than 50MB to only 23.4MB.

We also improved the javadoc and basic comments within the code. In particular, we focussed on carefully commenting code in major classes such as PredictionManager (model/PredictionManager) or the classifier (model/PredictionClassifier). Furthermore, we wrote a README file in order to explain how to run the application.

7. Coding intensity

The following graphic (downloaded from the SmartSurfPredictor repository on GitHub) illustrates the peaks of the three milestones:

Aug 16, 2015 – Oct 31, 2015

Contributions: c

Contributions to master, excluding merge commits

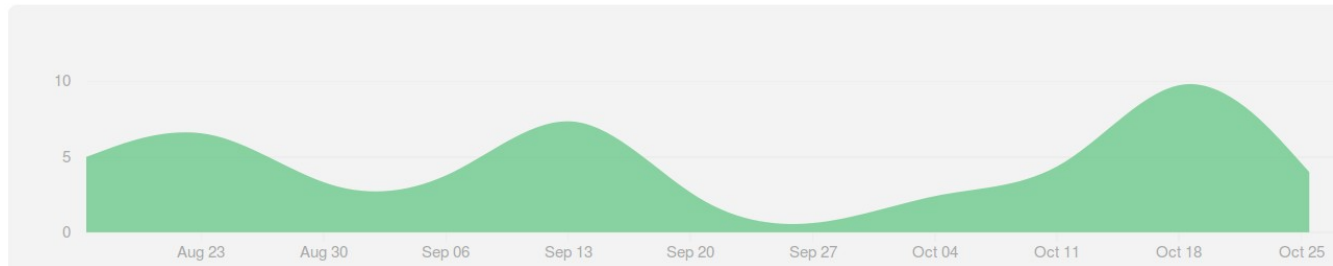


Illustration 1: Coding curve (GitHub)

2.3 Team spirit and team organization

We decided to use an agile software development methodology, similar to Scrum, with monthly milestones and weekly sprints.

A lot of the communication went over e-mails, as well as weekly group meetings. The meetings were about discussing the latest changes, proposing improvements and planning the next sprint. Before the meetings, each group member prepared a check list with topics and details to be discussed.

We also used the GitHub wiki in order to exchange links and file templates.

Within the code we used javadoc comments for describing methods and classes. Before implementing a module, we agreed on interface definitions. It should be crystal clear what output to expect and what input is necessary to use a certain method/module. This worked really well and the separation of the implementation went mostly smoothly. Issues and minor misunderstandings could be resolved in the weekly sprint meetings.

2.4 Tools

We used the Eclipse IDE and git as version control system. The repository resides on GitHub.

At the beginning, we used a branching strategy where each contributor had an own branch. These branches were regularly merged with a development (“dev”) branch. For convenience, this was later changed into a simpler two-branches strategy with one master and one dev branch.

Models and diagrams (UML class and sequence diagrams) were designed on Lucidchart, text documents on Google Drive.

2.5 Improving intelligence and feedback learning

After Milestone 1 the classifier worked, but it did not generalize well to unforeseen forecasts. For example, it sometimes suggested to go surfing in high swell (> 8 ft), supposedly because the training set did not contain such an example, labelled as “no”.

1. Improving sample data set

We solved this issue by firstly creating a better sample data set with a broad range of values for each feature. When setting-up the application, the user rates forecasts from that data set.

2. Classifier evaluation

Secondly, we evaluated various classifiers:

	Naive Bayes	SimpleLogistic	Multilayer Perceptron (NN)	RandomTree
Correctly classified [%]	76.2	88.1	90.5	83
Precision	0.79	0.88	0.91	0.84
Recall	0.76	0.88	0.91	0.83
F1-Measure	0.76	0.88	0.91	0.83
Time to build* [s]	0	0.2	0.3	0

Table 2: Results for dataset 3 (84 instances: 42 yes, 42 no)

* rounded to one decimal place

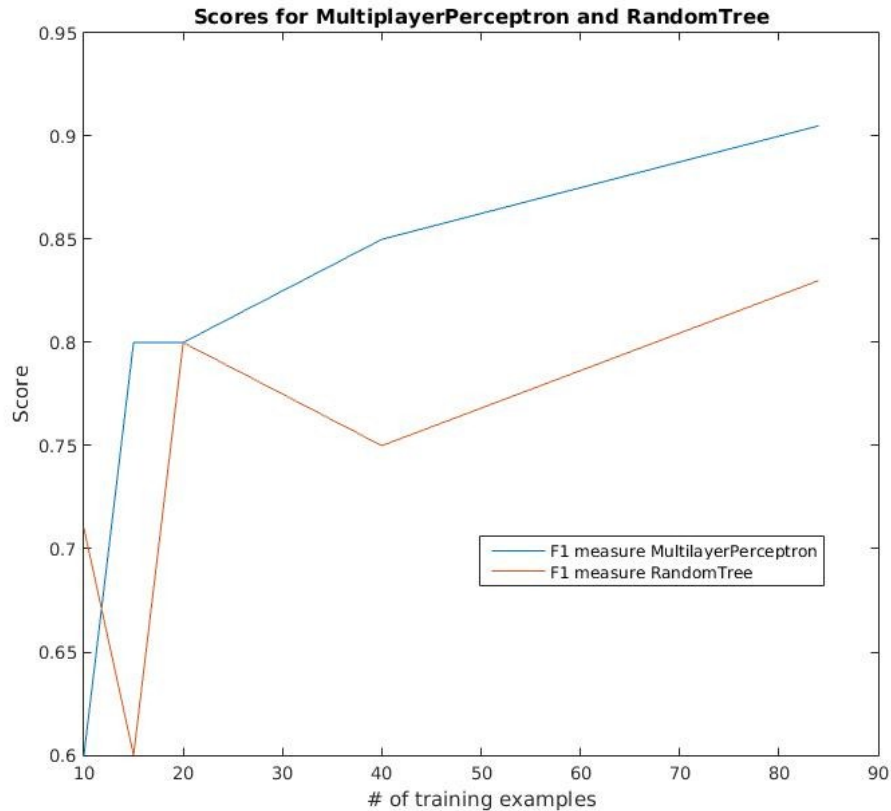


Illustration 2: Classifier scores

The Multilayer Perceptron delivered very good results, even when only trained with 20 training examples. It also turned out to be the slowest algorithm. It takes much longer to train a neural network than a decision tree. The forward and back propagation, as well as the gradient computation used to find the optimal parameter weights for the neural network are computationally much more expensive.

Still we decided to use a NN because the application would be running locally and would only have a single user at a time. Run-time performance would not a big issue for the Demonstrator and an accurate classification is the most important aspect. On the other hand, for scaling the application, we recommend a further evaluation of classifiers using more datasets created by a broad range of people.

3. Feedback learning

When setting-up the application, the user rates at least 20 forecasts. This dataset is used for the initial training of the classifier. From then on, each time the user receives forecasts, it is possible to enhance the training set by rating the live forecasts. The rated forecasts are appended to the training set. Before rating the next set of forecasts, the classifier is re-trained with the updated training set. It is not mandatory to rate the live forecasts. This is why the “User Feedback”-box below is drawn with a dotted line.

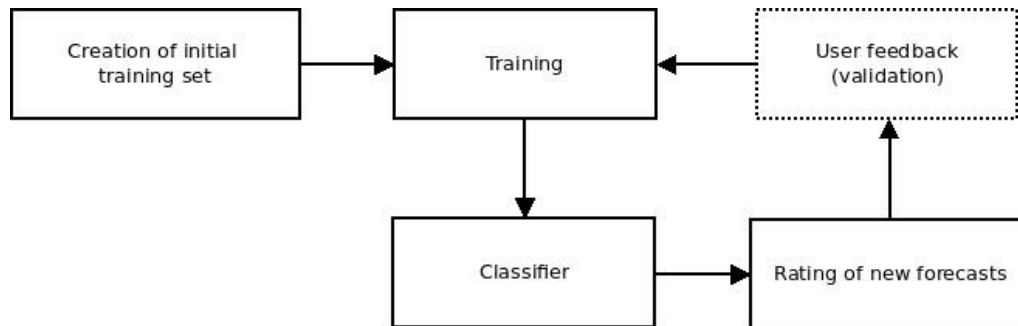


Illustration 3: Feedback learning

3. Operation Manual

3.1 General information

SmartSurfPredictor requires a valid magicseaweed API key in order to retrieve live forecast information. This API key has not been included in the GitHub repository in order to prevent misuse. For running the application, please use the provided ssp.jar file.

Comment for the lecturer:

The complete source code has been provided to you via Google Drive (link in e-mail from 29.10.15).

3.2 How to install SmartSurfPredictor on your computer.

The source code and an executable jar for SmartSurfPredictor is publicly available online:

<https://github.com/mbbuehler/SmartSurfPredictor>

For starting the Demonstrator, run "java -jar ssp.jar".

For running the application, please download the compiled source code from GitHub.

The GitHub repository includes a README, helping users to get started:

```
README.m

>>> SmartSurfPredictor <<<
Version: 1.3 (28.10.15)

--- Authors ---
Eeswari Avudainathan, eeswari91@gmail.com
Marcel Buehler, marcelchristoph.buehler@student.uts.edu.au

--- How to run application ---
run ssp.jar file, e.g.
java -jar ssp.jar
(Unix environment)

--- Training Set ---
An example training set has been provided. In case you would like to generate
your own training set, rename / delete the folder "user_data" and re-run the
application. The "Get Started"-Wizard will guide you through all necessary
steps.

--- Requirements ---
o JRE
o Internet (for fetching live forecasts)

--- Please Note ---
If you try to retrieve classified forecasts without having created a training
set of at least 20 examples or you did not select your favourite spots, you
will be asked to complete a proper set-up first. We recommend using the "Get
Started" Wizard. Thank you.

--- GitHub URL ---
https://github.com/mbbuehler/SmartSurfPredictor
```

Illustration 4: SmartSurfPredictor README

3.3 Welcome to SmartSurfPredictor: main view

This Java application has four main features.

- The first feature “Get Started” is a quick set-up wizard.
- The second feature “Add/Change Surf Location” allows users to modify the surfing location at any time.
- The third feature “Rate more Forecasts” allows users to rate more forecast. This helps the classifier understand user preferences better.
- The fourth feature “show Forecasts” allows users to view surf forecast for all your favourite surf spots.
- The exit feature terminates the application.

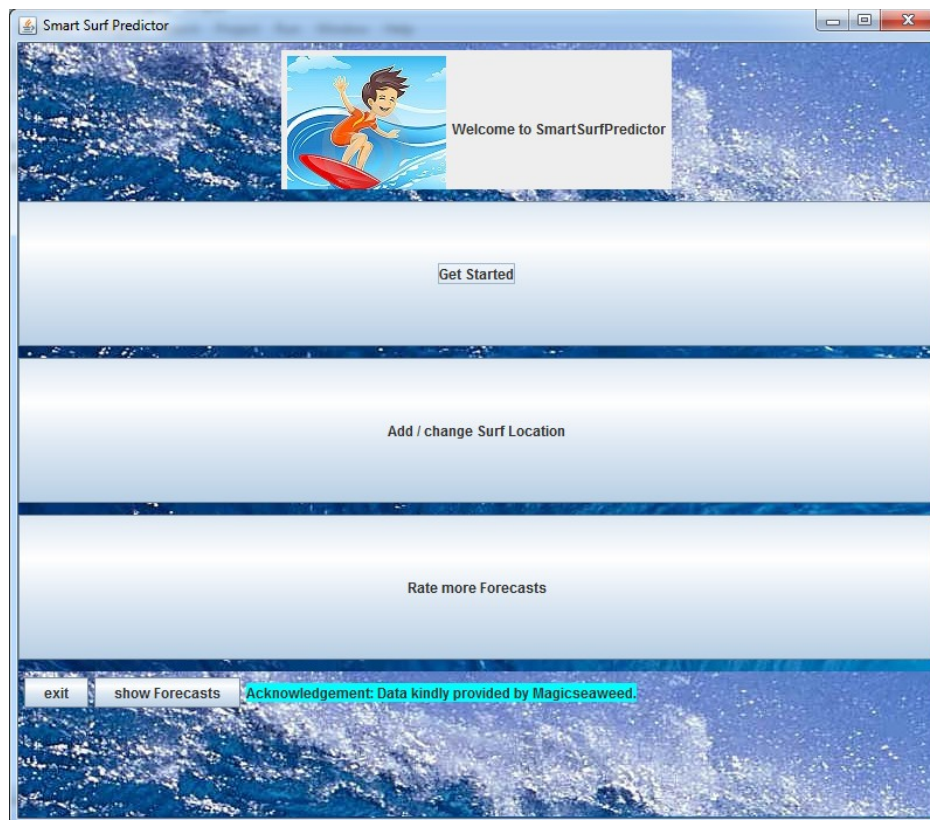


Illustration 5: SmartSurfPredictor's Main View

3.4 Set-up Wizard for SmartSurfPredictor.

When you select the “Get Started” feature on the main view, it directs you to this panel. This feature is a two-step process.

A) First step: Surf Location Selector

- Here, you get to select multiple surf location according to regions, states and surf locations by folding down the shift or Ctrl key on your keyboard while selecting. Click the “Submit” button to move on to the next step. The “Back” button brings you back to the main view (*Illustration 5: SmartSurfPredictor’s Main View*).
- This step has an error checking mechanism, it notifies you when you try to submit without selecting any region, state or location (*Illustration 6: Surf Location Selector (Step 1 of “Get Started”)*). It allows you to select again.

Illustration 6: Surf Location Selector (Step 1 of “Get Started”)



Illustration 7: Error checking mechanism for Surf Location Selector.

B) Rating Feedbacks, second Step of "Get Started".

For the classifier to initially understand your surfing preference, the second step requires you to rate 20 feedbacks (you only have to do this once).

- To rate feedbacks, you can read all the surfing information provided, see *Illustration 8: Initial forecast rating (Step 2 of "Get Started")* and then rate by clicking "Yes" or "No" button based on your preference.

Example: Based on *Illustration 8: Initial forecast rating (Step 2 of "Get Started")*, I would rate as "No", because I prefer a sunny weather and colder temperature.

- When you finish rating the first feedback, SmartSurfPredictor gets the next 19 forecast for you to rate. Once you complete rating 20 initial feedback, SmartSurfPredictor would save and pass your ratings to the classifier and return to the main view.
- You can hit the "Back" button any time to return to the main view. It would pass all your ratings to the classifier, before returning to the main view.

Without initially providing 20 feedback, you cannot view or receive surf notifications ("Forecast Feedback").

Your ratings are stored and analysed by the classifier immediately to understand your surfing preferences. This completes the "Get Started" feature.

Quick Feedback

Surf Prediction Feedback

Minumum Height (ft): 3

Maximum Height (ft): 5

Wave Ratings: ★ ★ ★ ★

Primary Swell Height (ft): 6.5

Primary Swell Rate (seconds): 12

Primary Swell Direction: N

Wind Speed (mph): 3

Wind Direction: S

Weather: ☽

Temperature (celsius): 24.0

Do you like this prediction ?

Acknowledgement: Data kindly provided by Magicseaweed.

Illustration 8: Initial forecast rating (Step 2 of "Get Started")

3.5 Second feature on main view : “Add/Change Surf Locations”

This feature allows you to modify your surfing location any time.

- Here, you get to select multiple surf location according to regions, states and surf locations by pressing the shift or Ctrl key on your keyboard while selecting (*Illustration 9: Add or change surf locations*).

Then click the “Submit” button for the application to update your preference and returns to main view (*Illustration 5: SmartSurfPredictor’s Main View*).

The “Back” button bring you back to the main view without saving changes (*Illustration 5: SmartSurfPredictor’s Main View*).

- This step has an error checking mechanism, it notifies you when you try to submit without selecting any region, state or location (*Illustration 7: Error checking mechanism for Surf Location Selector.*). It allows you to select again.

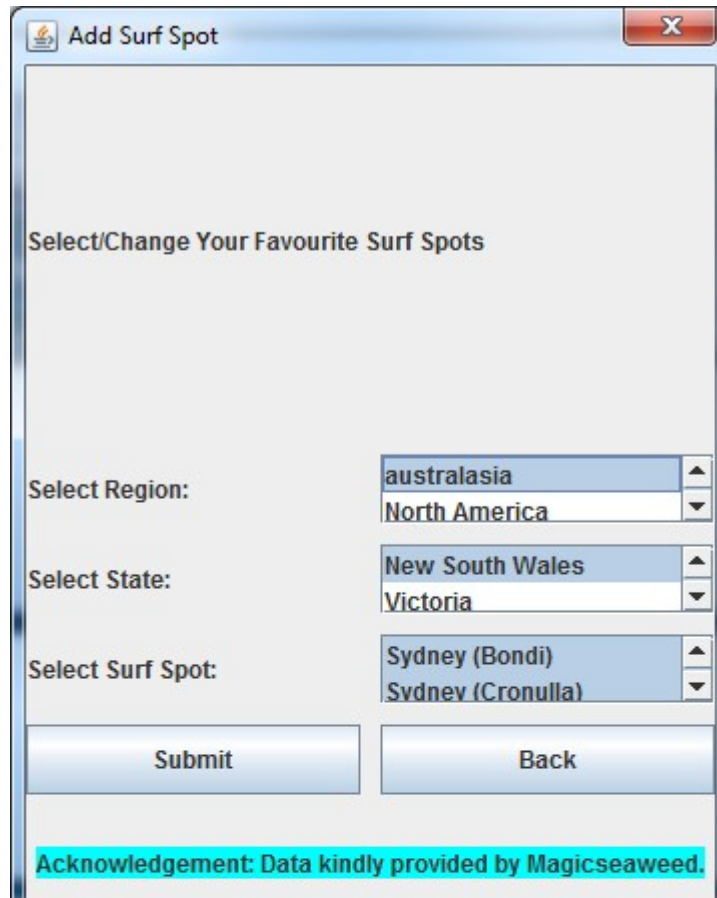


Illustration 9: Add or change surf locations

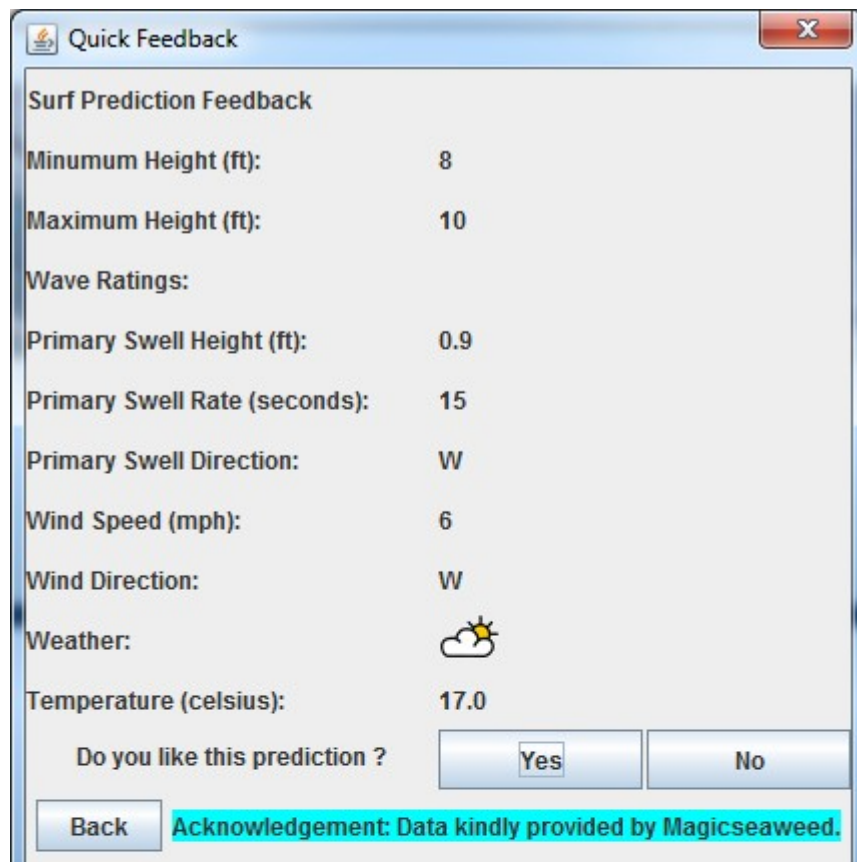
3.6 Second feature on main view : “Rate more Forecast”

This feature allows you to rate more forecasts in order to make the classifier to understand your surfing preferences better. This is optional but SmartSurfPredictor highly recommends you to rate more forecast regularly.

To rate more feedbacks, you can read all the surfing information provided like in *Illustration 10: "Rate more forecasts" feature* and then rate by clicking “Yes” or “No” button based on your preference.


Example: Based on *Illustration 10: "Rate more forecasts" feature*, I would rate as “No”, because the wave height between 8-10 feet, is too high for me.

- When you finish rating the first feedback, SmartSurfPredictor gets the next forecast for you to rate.
- You can hit the “Back” button any time, which will return to the main view. It would pass all your ratings to the classifier, before returning to main view (*Illustration 5: SmartSurfPredictor’s Main View*).



Quick Feedback

Surf Prediction Feedback

Minumum Height (ft):	8
Maximum Height (ft):	10
Wave Ratings:	
Primary Swell Height (ft):	0.9
Primary Swell Rate (seconds):	15
Primary Swell Direction:	W
Wind Speed (mph):	6
Wind Direction:	W
Weather:	
Temperature (celsius):	17.0

Do you like this prediction ?

Acknowledgement: Data kindly provided by Magicseaweed.

Illustration 10: "Rate more forecasts" feature

3.7 The last feature on main view: “Show Forecast”

This feature automatically pops up on your screen (*Illustration 11: Live forecast pop-up*) at 9AM and 3PM everyday and brings you the latest updates on all your selected surf location. If you wish to receive this information at any other time of the day, you can easily go to the main view (*Illustration 5: SmartSurfPredictor’s Main View*) and click on the “Show Forecast” button. You can receive all the surf forecasts here.

- The right side of the panel shows all your selected surf spots. You can click on the names to display the forecast details. The selected name would be highlighted in blue colour.
- The left side in *Illustration 11: Live forecast pop-up* is a black box, the information would dynamically change to show the selected surf spot forecast.
- Next to the surf spot on the left and information for “Surfing condition are:” on the right side, there is either a green image or red image. It is circled in yellow in *Illustration 11: Live forecast pop-up*.

The green image means the classifier has predicted that the surfing conditions for this location would meet your surfing preferences.

The red image means the classifier has predicted that the surfing conditions for this location would not meet your surfing preference.

- As you receive your forecast for each spot automatically twice a day, you can easily rate it by clicking the “Yes” or “No” button every time you receive this notification. This helps the classifier to constantly learn about you, even if you don’t use the “rate more forecast” feature on the main view.

When you rate, the application thanks you for your rating. Then, it shows you the next forecast based on your favourite surf spots (left side in *Illustration 11: Live forecast pop-up*).

Example: Based on *Illustration 11: Live forecast pop-up*, the application predicts that Bondi Beach has perfect surfing conditions, which meet my preferences. Therefore, the application indicates it with a green image. After checking on the forecast information in the black box, I agree with the classifier’s prediction. Therefore, I rate it as “Yes”. The classifier is predicting the surfing condition at the other selected spots are not suitable, so it displays a red image.

- You can hit the “Back” button any time in order to return to the main view. It would pass all your ratings to the classifier, before returning to the main view (*Illustration 5: SmartSurfPredictor’s Main View*).

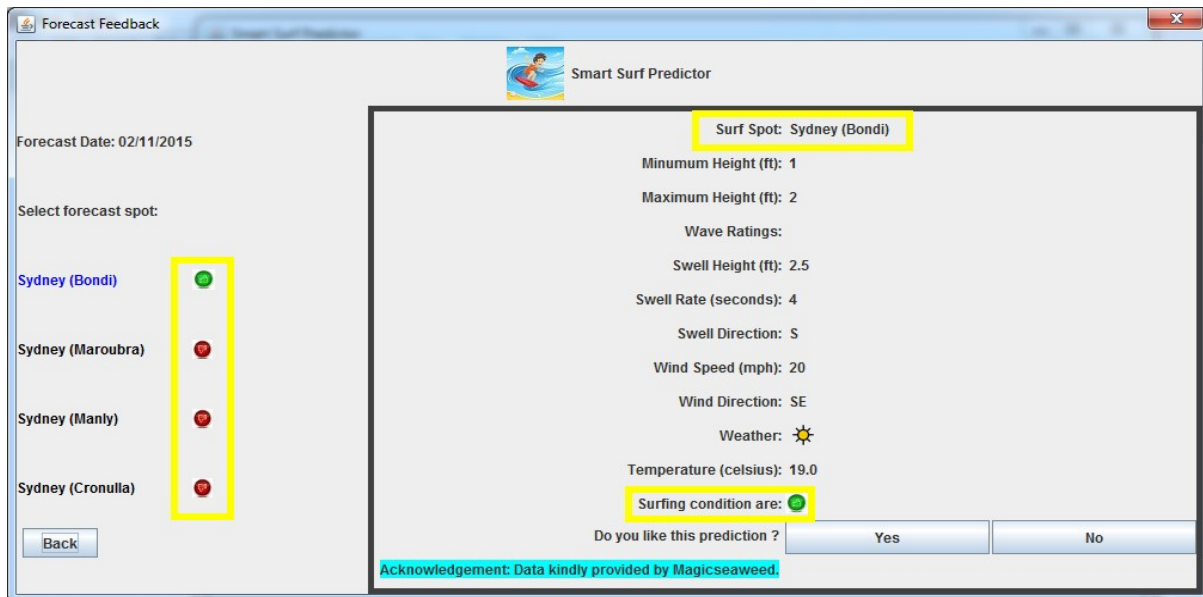


Illustration 11: Live forecast pop-up

3.8 Error Checking Mechanism for “Show Forecast” feature

If you do not select any surf spots or initially do not rate 20 forecast for the classifier to learn about you (you only have to do this once), then the application will not allow you to access the of “Show Forecast” feature and it won’t automatically pop up on your screen at 9AM and 3PM. The application will warn you about your mistake and shut down.

This is because the classifier will not be able to predict forecast ratings for you without initial data. Here you can see the error checking mechanism:

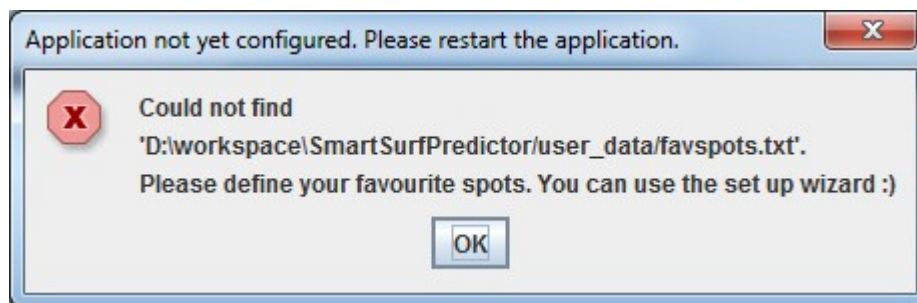


Illustration 12: You did not select any surf spots.

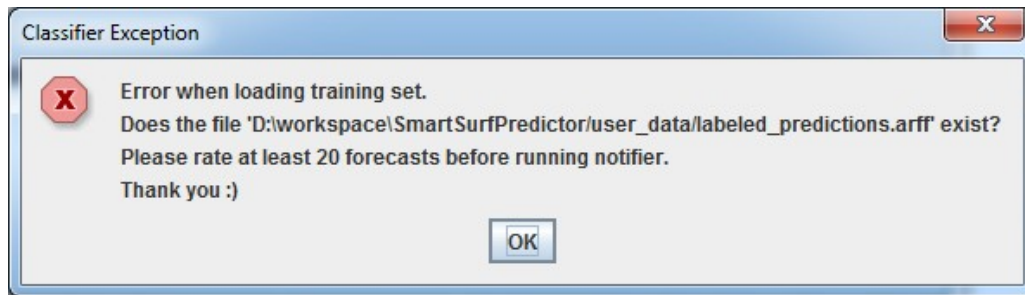


Illustration 13: You did not rate 20 forecasts.

3.9 Console Output

The console information indicates what the application is doing. This information is useful for debugging purposes.

```
>>> SmartSurfPredictor started <<<
> Application initialised.
> Loading user spots...
> Querying forecasts...
> Preparing data for classifier...
> Classifying...
> Training classifier...
> Classifier trained in 0.43 seconds.
> Aggregating scores...
> Displaying notification popup...
> Exiting...
>>> Thank you for using SmartSurfPredictor <<<
```

Illustration 14: Console output

4. Appendix

4.1 Class diagram: Forecast data modelling

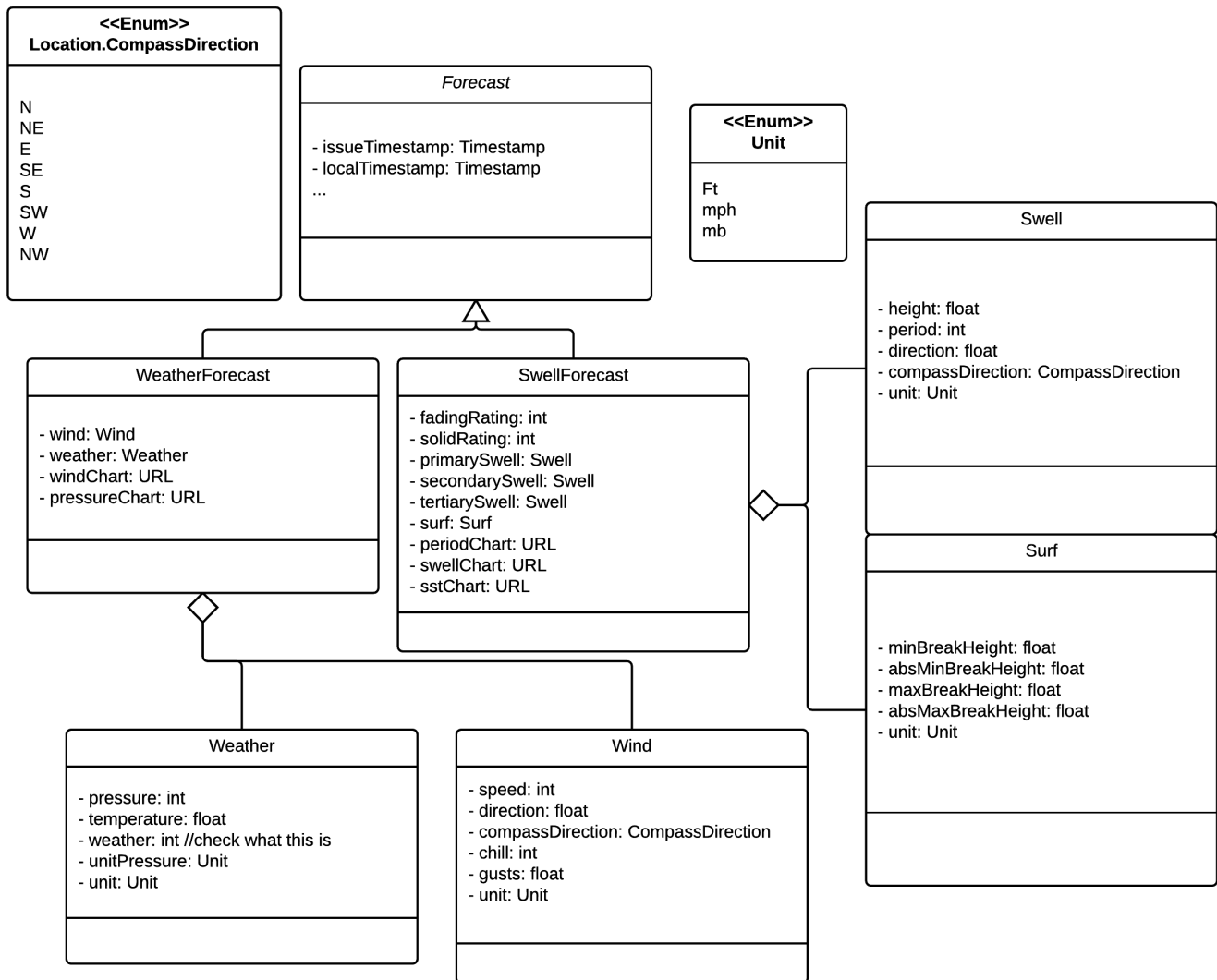


Illustration 15: SSP class diagram (forecast data modelling only)

When retrieving the JSON response from msw, the data is converted into an instance of the abstract class **Forecast**. Its relation to the subclasses **WeatherForecast** and **SwellForecast** and the various model classes is illustrated here.

4.2 Overview of file names and locations

1. System files

System files are static files that do not change for each user. System files are integrated in the exported .jar file and not directly visible to the end user.

File path prg_res/[name]	Description
spots.txt	Contains all available spots in the following format: <spotId>,<region>,<state>,<spotName>
unlabeled_sample_predictions.csv	Contains a balanced set of sample predictions that are used for the initial creation of the training set.

Table 3: overview files (prg_res)

2. User files

Files that are personalized for the user. When setting-up and using the application, these files will be generated on the fly.

File path user_data/[name]	Description
favspots.txt	Contains a list with the user's selected spots in the same format as "prg_res/spots.txt"
labeled_predictions.arff	Contains the training set for the classifier in the Attribute-Relation File Format
unlabeled_predictions.arff	Is a temporary file storing unrated live forecasts. The classifier loads the files from here and rates them, before displaying them to the user.

Table 4: overview files (user_data)

3. Test files

Test files are used for testing purposes only and are not exported to the .jar file.

File path test_data/[name]	Description
labeled_test.arff	Contains the test training set for testing the classifier.
unlabeled_test.arff	Is a temporary file storing unrated live forecasts that are used to test the classifier.

Table 5: overview files (test_data)

4.3 Attributes used for classifier

Attribute	Data type	Details
spotId	numeric	msw spot id
minBreakHeight	numeric	in ft
maxBreakHeight	numeric	in ft
fadedRating	numeric	msw rating (not personalized) ¹
solidRating	numeric	msw rating (not personalized)
primarySwellHeight	numeric	in ft
primarySwellPeriod	numeric	in ft
compassDirection	nominal: {N, NE, E, SE, S, SW, W, NW}	directions reduced to 8 possible values.
windSpeed	numeric	in mph
windDirection	nominal: {N, NE, E, SE, S, SW, W, NW}	directions reduced to 8 possible values.
weather	numeric	weather code (see msw website for details)
temperature	numeric	in degree C
show	nominal: {yes, no}	output class

Table 6: attributes considered by classifier

1 For further details check the msw website: <http://magicseaweed.com/developer/forecast-api> (visited: 1.11.15)

4.4 Sequence diagram

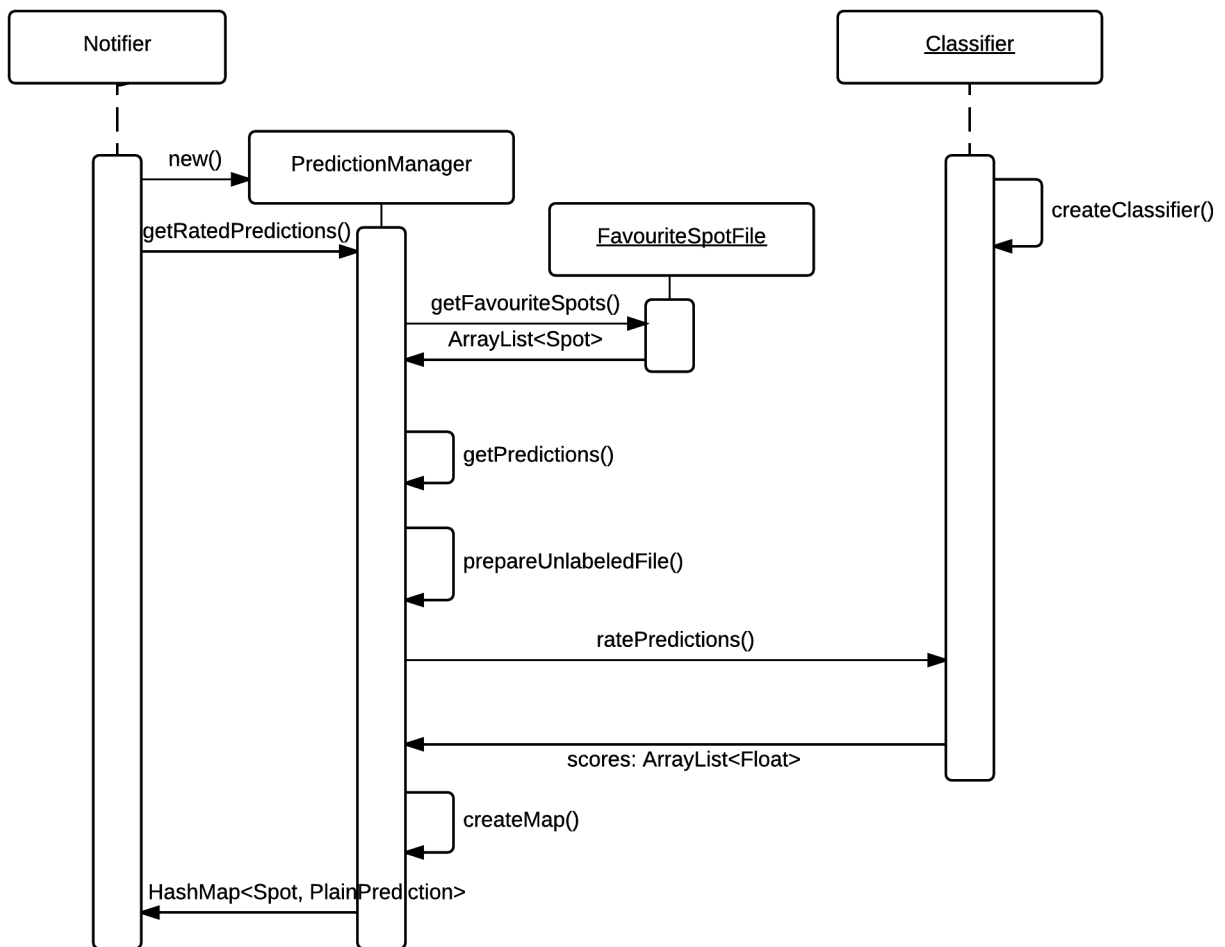


Illustration 16: sequence diagram for notification

The notifier starts by creating an instance of PredictionManager. The PredictionManager steers the work flow and handles most of the main method calls.

First, the user's favourite spots are read from the file system (getFavouriteSpots()). Then the live forecasts are fetched and converted into Java objects (getPredictions()). As this process involves many little steps of low importance for the understanding of the entire work flow, it is not modelled in this diagram.

In “prepareUnlabeledFile()”, the forecasts are prepared for the classifier and written to the file “user_data/unlabeled_predictions.arff”. Upon calling ratePredictions(), the classifier loads the unlabelled forecasts from the .arff file, computes their ratings and scores and returns the result as ArrayList.

Finally, the PredictionManager aggregates the result by packing the data into a HashMap. The HashMap contains a Spot object as key and PlainPrediction as value. PlainPrediction is a

flat representation of a Prediction (only containing primitive data types and Strings, no other objects). This should make it easier to extract the data in the GUI classes.

The HashMap is returned to the Notifier and passed further to the GUI classes for displaying the results to the user.

5. References / Links

Hall, M. et al., 2009, 'The Weka Data Mining Software: An Update', SIGKDD Explorations, Volume 11, Issue 1.

Abeel, T., de Peer, Y. V. & Saeys, Y. 2009, 'Java-ML: A Machine Learning Library, Journal of Machine Learning Research'

mvnrepository, 2015, 'Group: com.google.http-client', viewed 1 November 2015, <<http://mvnrepository.com/artifact/com.google.http-client>>.

mvnrepository, 2015, 'Jackson 2 Extensions To The Google HTTP Client Library For Java', viewed 1 November 2015, <<http://mvnrepository.com/artifact/com.google.http-client/google-http-client-jackson2>>.

magicseaweed, 2015, viewed 31 October 2015, <<http://magicseaweed.com/>>.

GitHub, 2015, 'SmartSurfPredictor', viewed 2 November 2015, <<https://github.com/mbbuehler/SmartSurfPredictor/>>.

Lucidchart, 2015, 'Diagrams Done Right', viewed 5 November 2015, <<https://www.lucidchart.com/>>.

Google, 2015, 'A safe place for all your files', viewed 5 November 2015, <<https://www.google.com/intl/en/drive/>>.