Enabling weather-based decision making for forestry pest and disease management: Visualisation Report

Connor McDonald
University of Pretoria
Department of Computer Science
u16040725@tuks.co.za

Gené Fourie
University of Pretoria
Department of Computer Science
u20797274@tuks.co.za

Keywords

Sirex noctilio, Leptocybe invasa, Forestry pest and disease management

1. VISUALISATION DISCUSSION

The visualisations for this project are built from the models developed and discussed in the previous Modelling assignment. The visualisations provide both a historical and future perspective of the effect of changing weather patterns and pest prevalence across South Africa. The historical perspective allows the user to analyse trends that may have led to a positive pest inspection. While the future perspective allows the user to extrapolate predictions across South Africa, including inaccessible regions covered in dense forest.

All visualisations included in this report are interactive and are available through the *streamlit* web application built for this project (discussed further in Section 2). The web application has four main pages: *Home*, *Prediction*, *Visualisation*, and *Upload*.

1.1. Historical Visualisation

1.1.1. Pest presence

Figure 1 is a snapshot of the historical presence of Sirex, which is deployed on the *Home* page of the web application. A similar map is also available for Leptocybe on the *Home* page. The visualisations allow the user to select '*All*' years of inspection, or a single year, to view the pest presence. These visualisations aim to allow users to perform their own exploratory data analysis to identify areas more prone to pests and spatially display the pest data used to build the models.

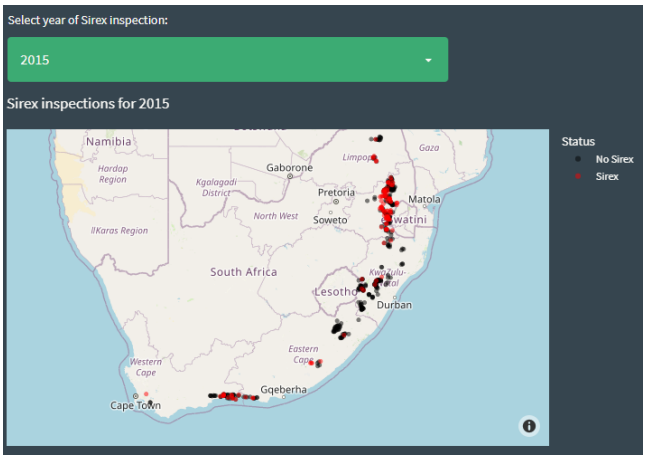

Figure 1: Sirex presence in 2015

1.1.2. Weather measurements

Figure 2 and Figure 3 are snapshots of the historical total rainfall and average temperatures, respectively, which are deployed on the *Visualisation* page of the web application. However, to generate the graphs, the user must enter the current month, total monthly rainfall, average maximum and minimum temperatures for the month, and an approximate location of where the measurements were taken. After running the model, the feature engineering script (similar to that explained in Section 1.3 of the Modelling report) is invoked to find the nearest active weather stations for the point location provided. Thereafter, the script retrieves the historical rainfall and temperature measurements from the relevant stations for the active period of the station. Note that the average measurements shown in Figure 8 (discussed in Section 1.3.1) are calculated using the results of this analysis.

Figure 2 and Figure 3 are then generated, which reflect the weather measurements specific to the location provided by the user. Additionally, the graphs allow the user to graphically view the change in rainfall and temperature trends compared to the actual rainfall and temperature (maximum and minimum) measurements for the provided location.

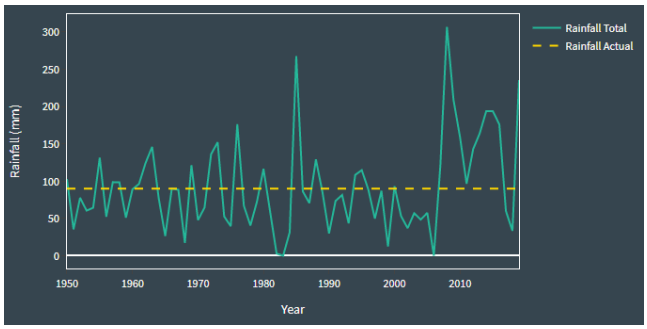

Figure 2: Rainfall trends for Feb at user-defined location

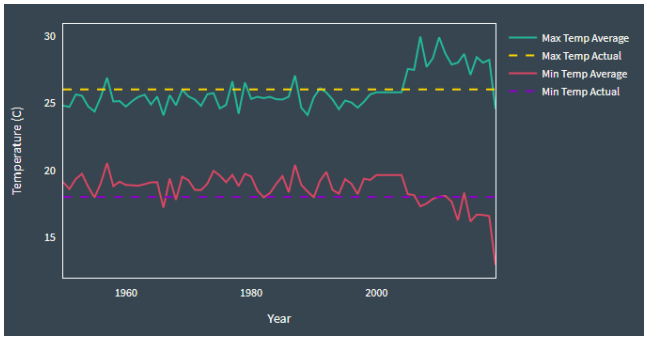

Figure 3: Temperature trends for Feb at user-defined location

1.1.3. Active weather stations

Figure 4 displays the distance of the nearest active weather stations to the user-defined location from 1950 to 2019. This visualisation aims to explain to the user that active stations change from month to month and year to year, as only stations with valid and complete measurements are used to determine the expected weather conditions at a location.

Additionally, the distance of the station to the user-defined location provides a qualitative indication of the applicability of the measurement. Note that a quantitative confidence analysis is futile given that other environmental factors, such as altitude, humidity, and wind, will affect local weather conditions.

Figure 4 shows that, in general, the stations used for rainfall measurements are in closer proximity to the point of interest than the stations used for temperature measurements. Additionally, the stations providing the measurements for rainfall (Station ID: 1012 at 20 km from the location) and temperature (Station ID: 1021 at 61 km from the location) are consistent before 2001. Thereafter, the active stations change, indicating that the prior stations were either decommissioned or complete data is unavailable for the subsequent period.

Additionally, Station ID: 5635 is found to be inactive in 2018, thus using the rainfall readings at Station ID: 5975 for the year, but active again in 2019, therefore, defaulting to the readings of the nearest active station (that is, Station ID: 5635). Overall. Figure 4 indicates that the user-defined location uses data from 6 unique stations over the period of consideration, with the furthest station located 109 km away from the point of inspection (Station ID: 5695 in 2019).

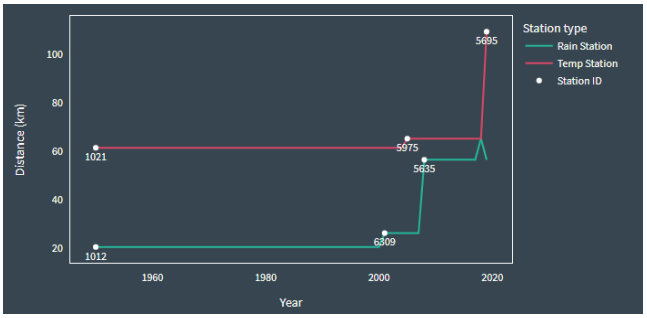

Figure 4: Distance from user-defined location to the nearest active station per year

Figure 5 spatially displays the stations identified in the analysis above against the user-defined location. The visualisation aims to provide confidence that the user-defined location is indeed used in the analysis and to show the proximity of the weather stations from Figure 4 above to the user-defined location. The map allows the user to hover over the station or inspection point (that is, the user-defined point) to show the Station ID and the straight-line distance to the inspection point. Figure 5 shows this feature for Station ID: 1021 (as discussed in Figure 4 above), located approximately 61 km away from the user-defined location.

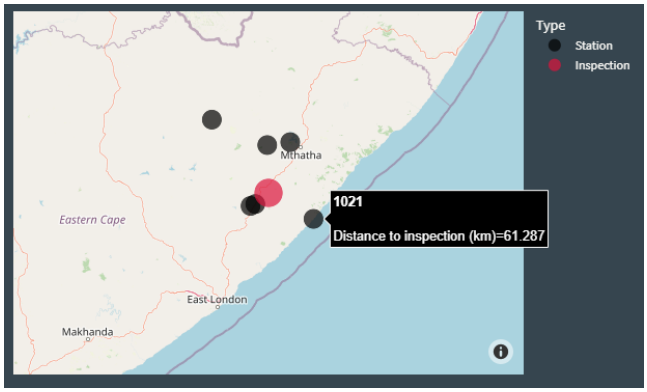

Figure 5: Nearest active stations

1.2. Model Visualisation

When navigating to the predictions tab of the application, the user is presented with numerous inputs such as rainfall, maximum temperature, minimum temperature, longitude and latitude and a map that shows the approximate location of the longitude and latitude points. The user can then select one of two pre-trained models, either XGBoost or Support Vector Machine. The inputs are fed into a feature engineering function that transforms them into 13 unique features. These features are used as inputs by the model to predict both pests simultaneously. It is important to note that each pest has a dedicated model, so when a user selects XGBoost, two pre-trained XGBoost models will be loaded from files, one for the Leptocybe pest and one for the Sirex pest.

Once both models have finished classifying the input data, the final prediction is printed on-screen with its respective class probabilities, shown in Figure 6. This provides context behind predictions and allows the user to make their own judgement on borderline cases.

Since not all app users will have a technical background, we briefly explain the chosen algorithm and write out the final prediction with its probability. For example, *The model predicted a negative Leptocybe inspection with a 94% probability.*

Figure 6: Prediction results

Lastly, the user is shown the ROC-AUC curve of the chosen algorithm trained on each pest. The concepts behind the ROC-AUC curve are explained with simple terminology so that the user can compare algorithms and their ability to predict pest prevalence across the two pest species. An example of this is shown in Figure 7.

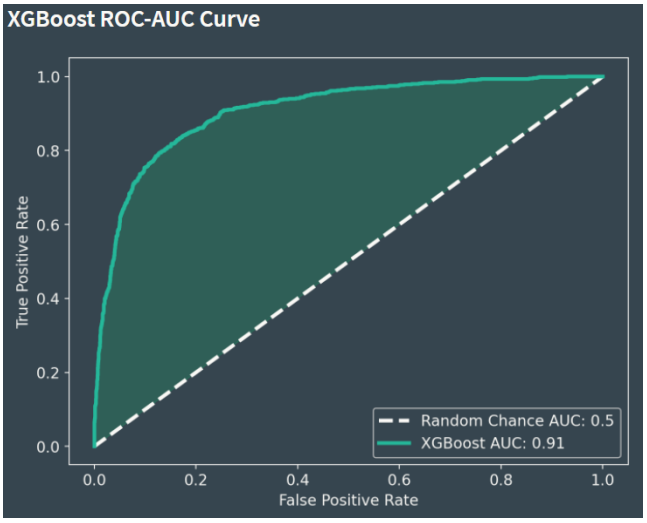

Figure 7: ROC-AUC curve of XGBoost model

1.3. Future Visualisation

1.3.1. At the user-defined point

Figure 8 provides a snapshot of Table 1 provided on the *Visualisation* page of the web application. The table displays the user's input measurements (actuals) on the *Prediction* page and the average monthly measurements for the user-defined location from 1950 to 2019. The table also indicates the difference between the actual and the average measurements. The table indicates that the measurements entered are indeed pulled through to the *Visualisation* page and show the comparison between the provided measurements and the average typically experienced for the user-defined location.

All parameters shown in the table, with the addition of the timeframe parameter, comprise of the input parameters for the prediction – where the timeframe parameter is calculated using the month and a prediction period provided by the user. Note that this transformation of parameters follows the same logic as the feature engineering process used to prepare the data to train the models (defined in Section 1.3 of the Modelling report). The measurements for the 'Average for location' in Figure 8 are derived from the analysis in Section 1.1.2.

	Total Rainfall (mm)	Average Maximum Temperature (C)	Average Minimum Temperature (C)
Input measurements	0.0000	20.0000	10.0000
Average for location	84.0000	29.7569	13.9758
Difference	-84.0000	-9.7569	-3.9758

Figure 8: Input parameters

Figure 9 provides a snapshot of the model results. This visualisation aims to repeat the results from the prediction page (discussed in Section 1.2 above) and allow the user to compare the results of the user-defined location to the predicted results across South Africa (further discussed in Section 1.3.2).

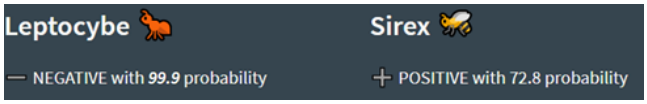

Figure 9: Repeat of prediction results

1.3.2. Extrapolation across South Africa

Figure 10 appears on the *Visualisation* page and shows the initial user-defined location and the positive prediction probabilities if the input measurements provided on the *Prediction* page are experienced across South Africa. A similar map is also available for Sirex on the *Visualisation* page. This visualisation enables the user to compare the prediction probability at the user-defined point and the prediction probability across the country if the same weather conditions are experienced.

Note that Figure 10 only shows the positive probabilities. For example, the probability of the grid block containing the user-defined point used in Figure 6 will be approximately equal to 12.99%. Each grid block shown in Figure 10 has dimensions of 50km by 50km, with 689 grid blocks used in the visualisation. The centroid of each grid block represents the location that is passed to the feature engineering script (as explained in Section 1.1.2). The spatial representation is produced using the following process:

1. The nearest active stations per centroid are identified, and their values are used to determine averages for the point.
2. The input measurements (actuals), the average measurements calculated above, and the timeframe parameter are used as the input parameters required for the prediction at the centroid.
3. The parameters are then used in the pre-trained models for Leptocybe and Sirex to determine the prediction per centroid and hence per grid block.

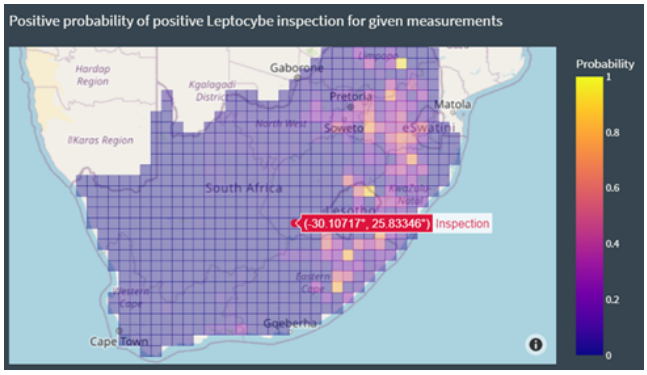

Figure 10: Prediction extrapolation

2. DEPLOYMENT DISCUSSION

The deployment of this project was done with a web application built with three components in mind: framework, user experience and backend processing. These components are discussed below.

2.1. Framework

This project was primarily done in *python* as it has powerful data analysis and machine learning libraries. For this reason, it was decided to build the app with *streamlit,* an open-source library

designed to build web apps with python backend architecture. Furthermore, streamlit has ample documentation and an active online support community. The app was built with a multipage framework to reduce the complexity and resource requirements of the code, as each page can be run independently instead of running the full app each time a page is refreshed. Furthermore, this allows the information in the app to be modularised where each page is dedicated to a topic.

Table 1: App Framework

Page	Description
Home	The home page is the default page of the app and provides the user with context behind the project by giving a brief description of what we aim to achieve with the app, what each pest looks like and lastly, where the pests have been historically sighted. The user is also given the option to download the previous reports, visit the partner's website, or see the source code with clickable buttons on the user interface.
Prediction	This page harnesses the models created in the previous assignment. It allows the user to interact with them through various inputs and ultimately make a binary prediction of pest presence. Since the app should serve a wide audience of different technical abilities, the predictions are explained with their probabilities. Lastly, the ROC-AUC curve of the model is shown and explained to help the user decide which model they should use.
Visualisation	This page builds on the user inputs specified on the prediction page and allows the user to interrogate their prediction. The page shows the historical weather patterns for the location selected and which stations were active during which period. Lastly, we use the selected model to make an extrapolated prediction across the country. This is visualised as a heat map of positive pest probability.
Upload	This page allows users to upload their records. FABI requested this as site surveyors need a tool to upload data collected during inspections.

2.2. User Experience

To improve the user experience on the app, we have integrated a session storing mechanism that stores variables and model outputs so that the user can navigate between pages without having to re-enter data or re-run models. Furthermore, we have built our visualisations with *Plotly, a* python library capable of producing interactive plots that can be displayed on web pages.

The models require 13 features generated by complex feature engineering processes. This would be too complex for the user to manually calculate and enter via the user inputs provided on the prediction page. Instead, we have narrowed this down to six inputs typically collected during a site inspection. Once the user has inputted these values, the feature engineering is invoked with background processes to generate the necessary 13 features. The user can then choose between an XGBoost model or an SVM model. This will then load two versions of the chosen model (one for each pest type) and make a prediction based on the inputs.

The inputs used to make the prediction are subsequently stored in a caching mechanism so that they can be used to generate the various plots on the visualisation page of the application. Here, we provide some historical weather data on the given location so that the user can identify anomalous patterns that may have led to pest infestation. Furthermore, we use the selected model to produce a probability heatmap of positive pest inspection across the country under the assumption that inputted weather characteristics are maintained across the country.

2.3 Backend Processing

To optimise the application, we stored models and their necessary scaling objects as files that could be read from the model directory when needed. This saves the user time as they no longer need to wait for models to train and for scalers to be fitted when making a prediction. This optimisation was also performed for the feature engineering of the 689 grid blocks (Figure 10), where the script was pre-run on the centroids to determine the active stations, hence the average measurements per centroid. Alternatively, a single interaction of the feature engineering script runs in the backend for every run of the model due to the unique location provided by the user. Furthermore, we implemented a caching mechanism designed by streamlit, which maintains the state of each page in the app. This allows the user to switch between pages without re-running any models or visualisation.

3. VALIDATION

3.1 Visualisation

To validate the visualisations developed in this application, we analysed the feature engineering process between the *Prediction* page and the *Visualisation* page to ensure that the input data was being correctly processed on both pages. This was essential for the heatmap visualisation shown in Figure 10 as it relies on this features engineering process to produce 689 predictions per map.

3.2 Deployment

To ensure that the app is used correctly, we have added multiple robustness measures that prevent users from entering inputs in a non-feasible range. For example, the maximum and minimum geographic coordinates correspond with maximum coordinates within South Africa. Therefore, it is impossible for users to make predictions or generate visualisations for points outside of South Africa's borders. While the model is capable of doing this, it would extrapolate the weather records from the nearest weather station as per the given dataset. This could lead to very misleading results if limits were not imposed on the input features.

4. NEXT STEPS

To fully realise the value of the application, the application needs to be integrated with Google BigQuery and the Partners' information hub platform. Fortunately, this is a relatively easy task as Google BigQuery has a dedicated API service which allows for seamless transfer of data between applications and data warehouses. The application has been built with a lightweight architecture, reducing the server requirements needed to host the application. Furthermore, it has been requested to create an input form on the partner's existing application, similar to that seen on the upload page of the application.