Predicting the Taxonomic Group of Venomous Animals Using Venom Proteins

Task 3: Report

Caleb Poock

Western Governors University

A. Project Highlights

Research Question or Organizational Need

The research question for this project was: "Can the taxonomic group of a venomous animal be predicted using only the protein composition of its venom?" This project also addresses an organizational need within the multidisciplinary field of venom research: namely, the need for data analysis solutions and tools, both for the field in general, and also specifically for the development of safer and more effective antivenom treatments.

Scope of Project

This project used the Tox-Prot dataset, transformed it from protein-level records to organism-level records, trained and tested a random forest model for predicting class based on distribution of protein families, and produced protein family distribution graphs for each class. The project did not cover further analysis, or the exact application to medical use. These are valuable pursuits for further research, but not within the scope of this project.

Overview of Project Solution

Methodology

This project used the CRISP-DM methodology, which involves six steps.

1. Business Understanding: The question and industry need stated above were identified, and the Tox-Prot database was discovered as a promising set of data for the project.
2. Data Understanding: The Tox-Prot dataset was downloaded, and the steps needed to transform the data for model training were identified.

3. Data Preparation: The transformations identified in the previous step were applied. These included simplifying protein family names, generating stand-in family names for certain proteins without assigned families, simplifying taxonomic lineage into six classes, and most importantly, pivoting the dataset so that each record was a single species instead of a single protein.
4. Modeling: The transformed dataset was split into 80% / 20% training and testing sets, respectively. A random forest classifier instance from scikit-learn was trained on the training data. A k-means clustering algorithm was also applied to the data, just to see if the six classes could be cleanly clustered by unsupervised learning, but the model struggled with distinguishing some of the classes. This attempt was left in the code, since the results may be useful for further analysis of venom similarity.
5. Evaluation: The random forest model was tested using the included .score() method on the test dataset, achieving an accuracy score of 95%.
6. Deployment: Informative charts about the protein family distribution in each class were generated with the matplotlib pyplot library. The entire project was made available on GitHub, and can be freely forked and customized for application. Both of these details provide information and tools for further research.

Tools and Environment

The language used for this project is Python 3. The project was developed in the VSCode IDE, and version control was handled using GitHub. The coding environment was handled by Conda within a WSL Ubuntu virtual machine. The Conda environment includes the following

dependencies: matplotlib, pandas, jupyterlab, scipy, numpy, scikit-learn, pip, and pickle. The primary code for the project was written a jupyter notebook .ipynb file, with libraries and classes from the above modules imported ass necessary. Pandas was the main library used to explore, clean, and transform the raw data; scikit-learn for the training and testing of the random forest model; and matplotlib pyplot for generating visuals.

B. Project Execution

Goals, Objectives, & Deliverables

The goals, objectives, and deliverables did not change from the proposal. No major issues with data transformation, model training, or model testing were encountered, and the appropriate graphs were generated satisfactorily.

Goal: To determine whether venom composition can be used to identify animal groups.

Objective: Transform the dataset to a useable form.

Deliverable: Standardize protein family names for grouping of proteins.

Deliverable: Simplify taxonomic lineage into 6 classes (Snakes, Spiders, Scorpions, Insects, Jellyfish + Relatives, Cone Snails).

Deliverable: Pivot table from protein-level records to organism-level records, with each protein family as a separate column feature.

Objective: Predict placement in classes based on venom composition.

Deliverable: Trained and tested random forest model.

Deliverable: Accuracy score of model.

Objective: Show distribution of protein families in each class.

Deliverable: A graph for each class showing the protein families present in that class, and the rates at which each family is present.

Methodology

As explained in part A, the CRISP-DM methodology was used to plan and execute the project steps. All steps followed the set plan without major deviances, and the model performed above 90% accuracy, so examination into a failing accuracy score was not necessary.

Timeline & Milestones

The data analysis and coding sections of the project (from "Establish Project Question and Identify Dataset" to "Generating Charts" in the proposal timeline) proceeded according to schedule. Flexibility was built into the schedule to account for days where no work would be done due to other commitments, and there were no unexpected events leading to loss of planned work time. The project proposal was submitted on 12/11/25, two days ahead of schedule. At the time of writing this section, this report and the accompanying video are on track to be finished by 12/13 or 12/14, one to two days ahead of schedule.

C. Data Collection Process

Data Collection

The data had already been selected and collected at the time the proposal was written, so the data collection method did not change. The .tsv file downloaded from UniProt did not present any problems when imported into a pandas dataframe.

Data Cleaning

539 records were missing protein family assignments, which were necessary for the model training. A few artificial protein family names were able to be generated based on shared names of certain proteins, but 515 of the 539 records did not fit into these generated families and were dropped. Once the data was transformed, an "Other" class of 31 records that did not fit into the six major classes was dropped, reducing the transformed dataset from 830 species records to 799 species records. No other issues were encountered.

Data Governance

No problems were encountered in using GitHub for version control. Each commit was pushed successfully and no issues in syncing occurred.

C.1 Advantages and Limitations of Data Set

Advantages: The Tox-Prot database is one of the most complete venom protein databases in existence. As Zancolli et al. (2024) explains, most other databases are focused on a single class of organisms; Tox-Prot is one of the only ones that could be easily used for the cross-taxon prediction that this project demonstrates.

Disadvantages: Since the dataset represents an aggregate record of the proteins found in each species, it only contains one record of each protein found in a species' venom. This makes predicting down to the species impossible, since only one record for each species exists in the pivoted data, which is why only the six overarching taxonomic classes were used in this project. A much larger dataset with multiple entries from individuals in each species would provide

better information on the venom composition variances within a species, and would allow for more granular prediction (if not to the species level, then potentially at least genus or family). The data would still have to be separated into the six classes used in this project, so it is a good first step in that direction. It should be noted that, to the author's knowledge, no such dataset exists, and would require significant research to generate.

D. Data Extraction and Preparation

Data extraction required minimal effort: the UniProt site has a download option to download data in a variety of formats (see here for the full dataset and download options). The .tsv format was selected, then read into a pandas dataframe using pandas.read_csv. Preparation was more difficult. The original dataset was structured with one record for each protein. There were four columns that were useful: Protein names, Organism, Protein families, and Taxonomic lineage. Protein names gave the primary name for the protein, as well as any alternative names. Organism listed the common name and Latin name of the organism that the protein came from. Protein families contained the hierarchical classification of the protein (superfamily, family, subfamily, etc.). Taxonomic lineage showed the classification of the species, from domain all the way down to genus. Organism, Protein families, and Taxonomic lineage all required transformation for the final dataset, while Protein names was used to generate a few artificial protein families for some of the records that had no assigned protein family.

Transformation of the dataset used a series of pandas as dataframes with subsequent transformations, and proceeded as follows: the Protein families column was simplified to the family level, cutting off distinctions at the sub-family level or below. This reduced the number of unique protein families from 608 to 308. Another eight protein families were generated in order

to retain some of the proteins that had no official family, bringing the total number of protein families to 316. These were generated manually, by examining the protein names and grouping proteins with identical names but different identifiers (e.g. grouping “Ampulexin 1”, “Ampulexin 2”, and “Ampulexin 3” together in the artificial protein family “Ampulexin”). Initially, 21 protein families were generated in this manner, but only eight were found to be useful; the others were groupings that did not represent similarity between proteins (such as proteins that were named after the organism they came from, instead of their function, or naming conventions used to indicate incomplete data). These eight families only contained 24 proteins out of 539 with no family name; the other 515 were dropped from the dataset.

The next step was to pivot the dataframe. This involved extracting the Organism and Protein families columns and extending the “Protein families” column into a series of columns for each protein family. This pivot combined each record from a single species into one record, with the species name as the index, and a 0 or 1 in each protein family column, representing the absence or presence of that protein family in the organism’s venom.

Once the data was pivoted to organism-level records, the Taxonomic lineage column had to be simplified into the six classes that would be used as labels for the random forest model. This involved searching the lineage of each organism for the name of the class, then putting that name in a new generated column that would be used for the predictive labels. The six classes were Serpentes (snakes), Scorpiones (scorpions), Araneae (spiders), Insecta (insects), Cnidaria (jellyfish and relatives), and Conoidea (marine cone snails). 31 of the 830 records in the pivoted dataframe did not fall into one of these six classes, and were dropped (these included platypuses, centipedes, shrews, and a few other venomous animals).

The final dataframe looked like this:

Organism (index)	**Protein family A (feature)**	**…**	**Protein family Z (feature)**	**Taxon (class / label)**
Latin name (Common name)	0	…	1	Class

This features-label structure is standard for training and testing machine learning models; the training or testing function is passed the features as the X (data) argument and the class column as the y (labels) argument. As a final step of data preparation, 20% of the dataframe was split off into a test set, with the other 80% used for training. The split was stratified, ensuring that each class was represented at accurate frequencies in the test set.

E. Data Analysis Process

E.1 Data Analysis Methods

This project used a random forest classifier model from the scikit-learn library. The random forest algorithm is well-suited for multi-class classification tasks (this project involves 6 classes), since it breaks down the classification into multiple smaller decision trees. This allows it to handle the complexity of predicting more than two classes, and having multiple smaller voting units helps avoid overfitting the data. It also handles imbalanced datasets well, which was helpful since the Serpentes class represented over 30% of the dataset and the Cnidaria class represented just over 8%.

Additionally, scikit-learn provides a number of methods to investigate the performance and output of its models, which allows for easy evaluation of feature importance, accuracy, and other metrics. These investigative tools prevents the model from being a “black box” where the

internal process is not understood, and allows for easier investigation when performance does not match expectations. For all these reasons, random forest was an excellent fit for this project.

E.2 Advantages and Limitations of Tools and Techniques

Python and Imported Libraries

Advantages: Python is an easier language to learn and use than more complex languages. With the suite of freely available libraries, it is easy to import the exact functionality needed. The pandas library allows for in-depth data manipulation and analysis, while scikit-learn is a near-comprehensive resource for machine learning and associated methods. Matplotlib pyplot allows for easy generation of charts for visual depiction of data.

Disadvantages: Python is an interpreted language, which means it runs slower and the code must be evaluated by an interpreter every time it is run. For time consuming methods such as certain types of machine learning, this can lead to very long runtimes. Fortunately, the dataset in this project is relatively small; training and testing the model would have taken significantly longer with a larger dataset.

Jupyter Notebook

Advantages: Jupyter Notebook enables easy iteration of code and analysis. Short blocks of code can be run without needing to run the entire document's worth of code, allowing for fine-tuning.

Disadvantages: Variables do not persist between sessions, requiring re-running of code each time the project is opened, or storing of variables in documents (such as with the pickle library), which can be clunky to implement in large projects. The Jupyter Notebook kernel can

also crash from computationally intensive code or other errors, and it can be pretty opaque when it comes to troubleshooting these issues.

Random Forest Model

Advantages: The random forest algorithm handles multi-class classification well, and the ensemble method of aggregating multiple decision trees allows it to handle complexity without overfitting. The scikit-learn library also provides methods for investigating model performance and feature significance, among other metrics.

Disadvantages: This algorithm can be computationally intensive. The dataset used in this project is thankfully small (which comes with its own limitations); for any much larger datasets, the workflow in this project may take considerable time to run when training or testing the model.

E.3 Application of Analytical Methods

In order to train the random forest model used for analysis, the data had to be transformed into a set of features (protein families) and labels (taxon). The features are the set of data that the model uses to learn patterns about each record, and the labels tell it the correct prediction for that record. The full set of steps to get the data into this format can be found in part D of this document. Verifying the success of the data transformation was accomplished by visually examining the resultant dataframe and ensuring that the columns present were correct with the planned format.

Once the data was transformed properly, it was used to train and test the model. The data had to be split into two sections, one for training and one for testing. This is to ensure that the

data used to test the model is not involved in training the model; otherwise the model may not perform well on new data. To train the model, the test data was passed to the scikit-learn .fit() function. As long as this function runs without error, the model has been trained successfully, and no other verification of training is necessary. This is not, however, the same as verifying the model's performance or accuracy, which is the purpose of the next step, testing.

Testing used the scikit-learn .score() function, which runs the trained model on the test dataset and returns the aggregate accuracy score across all classes. After some optimization of model training parameters (mainly by increasing the number of estimators in the random forest), an accuracy score of 95% was achieved.

F Data Analysis Results

F.1 Statistical Significance

The model used in this project is an ensemble supervised classification model. The algorithm used was the random forest algorithm, which is composed of a number of individual decision trees. Each tree is trained on a subset of the data, and the final model makes predictions based on the votes from each of the component decision trees. Scikit-learn's .score() method was used to evaluate the model performance. This function calculates the accuracy of the model across all classes. For this project, the benchmark for the model to be considered successful is an accuracy score of 80% or better. The final accuracy score of the model was 95%; it only misclassified 15 out of the 799 records. The conclusion drawn from this high score is that the venom composition of disparate taxonomic groups is distinct enough that they can easily be accurately classified using only this information, which was the hypothesis for this project.

F.2 Practical Significance

The accuracy score of 95% is excellent, and far beyond the minimum threshold set for the model to be considered successful. While it would require a larger, more granular dataset to be fully applied in the real world, this result indicates that the model would be well-suited to the practical applications discussed, primarily the application of identifying which antivenom to administer in an emergency situation.

The practical application would look something like this: the model (or a similar one trained on a much larger volume of data) would be applied to a regional set of dangerous venomous animals. The feature importance of the model would be used to determine the crucial proteins useful for distinguishing between the relevant species. A medical test could then be developed that would quickly register which of these diagnostic proteins were present in a victim's blood, and medical professionals could use this information to accurately select the appropriate monovalent (single-species) antivenom. Since polyvalent (multi-species) antivenoms are associated with more frequent medical complications, this would reduce the rate of these complications, and could also reduce the cost of antivenoms in general, since monovalent antivenoms are easier to produce than polyvalent antivenoms.

F.3 Overall Success

All criteria for success outlined in section B6 of the proposal were met. The dataset was cleaned and transformed into the proper format without issue, the model was successfully trained, and it achieved an accuracy score well above 80%. This high accuracy is both a statistically significant result, and also indicative of the effectiveness of practical application to venom identification in emergency medical situations. The six protein distribution graphs were

also generated successfully and are clearly readable, giving an overview of the protein families present in each class and their relative proportions.

G. Conclusion

G.1 Summary of Conclusions

The 95% accuracy score for the trained random forest model supports the hypothesis that venom composition can be used to distinguish between disparate taxonomic groups of venomous animals. This result indicates that the model would be well-suited for the applications discussed in the proposal and this report. These include:

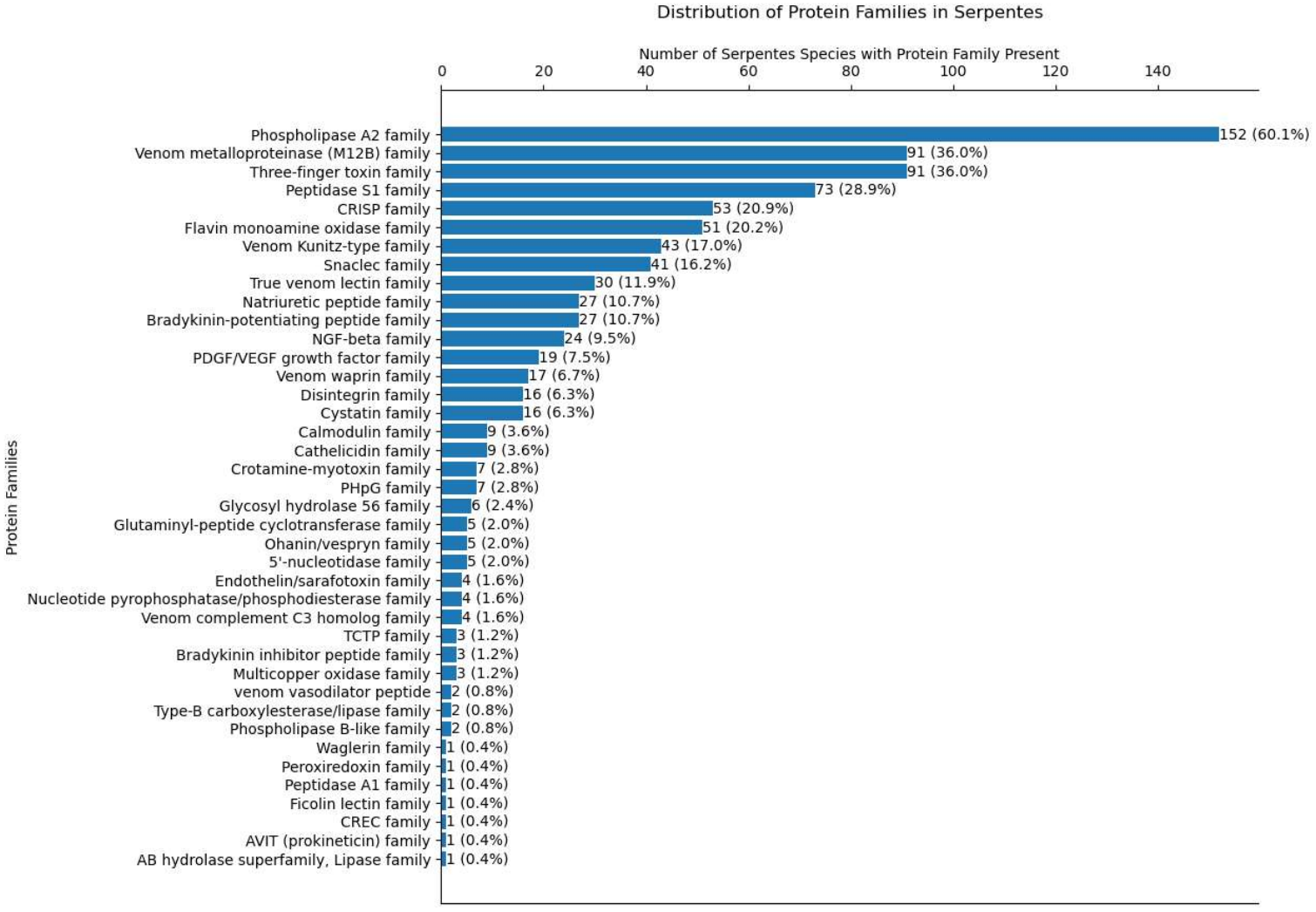
- The main practical application, identification of venom by composition for the purpose of administering the correct antivenom in medical emergencies.
- Use of the model as a tool for venom research using Tox-Prot, one of the largest collections of venom composition data.
- An argument for the usefulness of consolidated venom data and a step toward a unified suite of venom data and integrated analysis tools.

G.2 Effective Storytelling

The model had a high accuracy score, so visually examining the cases in which it failed is unlikely to be usefully informative. Since the hypothesis was that the venom compositions of the six taxonomic classes are different enough to be used for identification, it seemed proper to show visually the difference in composition between these classes. The matplotlib pyplot library presented an easy way to generate these graphs within the same Jupyter notebook that was used for the rest of the data processing and analysis. This allowed the avoidance of difficulties

involved in exporting the finished data and importing it to a different software to generate visuals. A simple function was written to generate the graphs, which required some fine-tuning to get the labels to display properly and the bars to be spaced so that each bar was separate and not blending into its neighbors. The initial graphs did not have percentages attached to each protein family, only raw counts, and this was found to make comparison between the classes difficult.

The final graphs are simple horizontal bar charts, with the names of protein families along the y-axis and the bars showing their level of presence within the class along the x-axis. An example of one of the charts is shown below:



These charts allow for two main analyses. The first is evaluating which protein families are most abundant within a class, which provides candidates for proteins that might be useful in developing the medical tests described in part F2. The second involves comparing protein family distribution between classes, in order to determine which protein families have overlap between classes. This second analysis is useful both in research for determining common venom targets, and also for supporting the first analysis by identifying protein families that would not be useful for medical diagnostics, since they are present in the venoms of multiple classes. For these analyses, the protein families with low abundances could be dropped to make the graphs easier to read, but for the purposes of this project, the full list of families present has been preserved for completeness.

G.3 Recommended Courses of Action

Recommendation 1: Given the accuracy of the model and the ease with which it was trained, medical technology companies should pursue using this workflow to inform the production of medical diagnostic tests. These tests could aid emergency medical personnel in correctly identifying the appropriate antivenom to administer, reducing injuries from both venomous animals and less-effective polyvalent (multi-species) antivenoms.

Recommendation 2: This project was completed in less than a month. The dataset was easy to use, and the model training was simple and quick. These factors are supportive of the argument that Zancolli et al. (2024) makes, that a centralized, consolidated, standardized database of venom data would be an invaluable resource for the venom research field. Organizations within this field should pursue and support the creation of such a resource, and

this effort should involve not only venom researchers, but also people familiar with data science and bioinformatics.

H Panopto Presentation

Panopto link to video (requires WGU login):

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=345e8f92-9753-4095-829b-b3b20168057f

References

Zancolli, G., von Reumont, B. M., Anderluh, G., Caliskan, F., Chiusano, M. L., Fröhlich, J., Hapeshi, E., Hempel, B.-F., Ikonomopoulou, M. P., Jungo, F., Marchot, P., de Farias, T. M., Modica, M. V., Moran, Y., Nalbantsoy, A., Procházka, J., Tarallo, A., Tonello, F., Vitorino, R., … Antunes, A. (2024). Web of venom: Exploration of big data resources in animal toxin research. GigaScience, 13. https://doi.org/10.1093/gigascience/giae054

Appendix A: Evidence of Completion

The Jupyter notebook used for the entirety of the project code is included in the submission documents as an html file. The notebook also includes the six generated bar charts. The GitHub repository is also public and viewable, however the author may continue to use these files to conduct further research and analysis, so the repository may contain more than is included in the submission, depending on the time of viewing. The repository can be viewed here: https://github.com/teqqqie/D502-Data-Analysis-Capstone

The full Tox-Prot dataset used for the project can be viewed or downloaded here: https://www.uniprot.org/uniprotkb/?query=taxonomy%5Fid:33208+AND+(cc%5Ftissue%5Fspecificity:venom+OR+cc%5Fscl%5Fterm:nematocyst)+AND+reviewed:true