School of Computer Science & Technologies

Field of study: Computer Science

Aysegul Akyuz
40698

PredictVet: AI-Supported Animal Disease Prediction System

Ruslana Prus

Warsaw, 2025

Contents

1. Basic information

Project name	PredictVet: AI-Supported Animal Disease Prediction System
Project goal	The aim of this research is to improve veterinary practice, monitor animal health to identify possible threats of disease at an early point, and improve the welfare of animals. At present, both large-scale agricultural businesses and individual pet owners face significant health and economic losses due to animal diseases. In addition, the prompt diagnosis of these diseases not only prevents extensive losses by stopping their spread but also improves the effectiveness of treatment. Conversely, traditional veterinary inspections tend to have a reactive approach, where treatment is initiated once the symptoms arise. This project seeks to establish a proactive approach through the use of data analytics and artificial intelligence. The test will analyze possible sickness risks by examining behavioral and biometric information gathered from the animals, including food consumed, body temperature, and physical activity. Big data sets will be utilized in training machine learning algorithms with a view to creating models with the potential for early detection of disease symptoms. This application will accelerate diagnosis, help veterinarians make decisions, and save costs by eliminating unnecessary tests. Further, when specifically used with livestock, the application of predictive measures will assist in disease containment and thus enhance herd health management. To enhance animal production and health management more effectively, the project will also involve creating a user-friendly application that can take animal data as input and provide disease predictions.
Brief description of the project	In order to improve animal production and health management, the project will also involve developing an intuitive application that allows users to enter animal data and receive disease forecasts. How to Proceed: 1. Data Acquisition and Preparation: Collect and prepare relevant data for animal diseases from reliable sources, i.e., Kaggle, to prepare it for use by machine learning models. 2. Model Development: Utilize the provided animal data, implement machine learning models (Random Forest, XGBoost, and K-Nearest Neighbors) to predict the occurrence of diseases. 3. Training and Evaluation of the Model: Train machine learning models, assess their performance, and select top-performing models based on metrics such as recall, accuracy, and precision. 4. User Interface Development: Create a basic mobile or web application to allow users to enter animal data and receive disease predictions. 5. Model Validation: To guarantee system accuracy and reliability, validate predictions against actual data. 6. Final Analysis and Reporting: Interpret the results, review the effectiveness of the system, and prepare a comprehensive report that summarizes the project results and recommendations.
Competitor product analysis	Artificial intelligence (AI) technologies are increasingly being utilized in animal health management and monitoring. The overall goal of such systems is to monitor animal health, detect disease early, and improve welfare overall. The following is an overview of the various efforts and applications and the advantages and disadvantages associated with these technologies: 1. IntelliPig: Face-Based Pig Health Monitoring System[2] The IntelliPig project aims to monitor pigs in farm environments using facial recognition technology. This system collects and analyzes data such as health status, feed and water consumption by recognizing individual pig identities. This makes it possible to detect early disease and create individual care plans. Advantages: - Early disease detection. - Individual health monitoring. Disadvantages: - The accuracy of facial recognition technology can be affected by environmental factors.

	2. Artificial Intelligence-Supported Veterinary Applications[3] Artificial intelligence applications developed for veterinarians are used in processes such as disease diagnosis, treatment planning and patient follow-up. These applications help veterinarians make faster and more accurate decisions with big data analysis and machine learning techniques. Advantages: - Faster and more accurate diagnoses. - Increased efficiency in the treatment process. Disadvantages: - It can be difficult to collect and analyze data correctly. - The lack of user-friendliness of the systems can make it difficult for veterinarians to adapt. 3. Analysis of Animal Emotional States with Artificial Intelligence[4] Researchers analyze the facial expressions of animals using artificial intelligence technologies and thus evaluate their emotional states. This method is used to monitor the stress levels, pain levels and general well-being of animals. Advantages: - Health and welfare assessment based on the emotional states of animals. - Improvements aimed at increasing the level of well-being. Disadvantages: - Facial expressions may not always reflect the correct emotional state. - Emotional analysis is more meaningful when supported only by physical health data. 4. Animal Behavior Monitoring with YOLOv8[5] Ultralytics' YOLOv8 algorithm is used to monitor and analyze animal behavior in real time. This system tracks the movements and activities of animals using non-invasive methods, thus providing information about their health status. Advantages: - Real-time and non-invasive monitoring. - Health status analysis based on animal movements. Disadvantages: - Environmental conditions (e.g. lack of light) may affect the results. - May require high processing power. 5. Farm Sense: Pig Behavior and Smell Monitoring System[6] The Farm Sense project uses machine vision and sensor technologies to monitor pig behavior and scent trails. This system monitors the activity, feed intake and general health of pigs, providing early disease detection and welfare assessment. Advantages: - Health status monitoring with both behavior and scent trails. - Early disease detection and welfare monitoring. Disadvantages: - May require high-cost sensors and technologies. - Sensor accuracy may be affected under farm conditions. 6. Journal of Toxicological Sciences[7] The Journal of Toxicological Sciences publishes research on animal health and safety. It includes studies on the effects of toxic substances that may be harmful to animals and the detection of these substances. Advantages: - Detection of substances that are harmful to animal health. - Safe production can be increased with toxicological studies. Disadvantages: - Detection of toxic substances requires complex analyses. - Focusing only on toxic substances can ignore all aspects of health status.

	7. FAO Animal Health Information Source[8] The Food and Agriculture Organization of the United Nations (FAO) provides global standards, guidelines and information resources on animal health. This platform supports international cooperation on the control and eradication of animal diseases. Advantages: - A comprehensive source of information on animal health on a global scale. - Contributes to international cooperation for disease control. Disadvantages: - Implementation processes must be adapted to local needs. - General information may need to be converted to more specialized data. 8. CDC One Health Initiative[9] The Centers for Disease Control and Prevention (CDC) examines the links between animal, human, and environmental health within the framework of the One Health approach. This initiative develops integrated strategies for monitoring and controlling zoonotic diseases. Advantages: - Early detection and control of zoonotic diseases. - Strengthening the link between animals, humans, and the environment in health management. Disadvantages: - Multidisciplinary approach may create difficulties in implementation. - Effective management of resources is necessary. Results and Competitive Product Analysis The implementation of various artificial intelligence technologies enhances health management greatly in animal health monitoring systems. Every approach, nevertheless, carries benefits and drawbacks of its own. Whereas, for instance, behavioral monitoring and facial recognition systems can yield accurate information on the health of an animal, the accuracy of the systems may be compromised by technological constraints and environmental issues. Whereas early diagnosis and treatment are enabled through AI-based systems, which also save costs and enhance efficiency, there are drawbacks in terms of upfront cost and adoption of the technology.[10] Capitalizing on the strengths of these current systems while taking into account their weaknesses can create a more efficient and affordable solution in creating a new, enhanced system. Through a more precise, cost-saving, and accessible animal health monitoring system, this solution can differentiate itself from other applications in the sector.[11]
List of technologies used	*Python 3.8+* *Scikit-learn 1.3.2+* *HTML/CSS/JavaScript* *Flask 2.0+* *Joblib* *Render.com*[13] *Power BI*[14]
Description of the technological stack and justification of selected technologies	**Connections Between Individual Technologies:** The PredictVet system is built upon a tightly integrated stack of technologies that support both machine learning and web deployment. At its foundation, **Python 3.8+** serves as the main programming language for data handling, model training, and backend logic. Within this environment, **Scikit-learn 1.3.2+** is used to build the disease prediction model. It also manages data preprocessing, including label encoding and multi-label binarization of symptoms. To enable real-time prediction and user interaction, the trained model is serialized using **Joblib**, allowing for efficient model loading during deployment. The **Flask 2.0+** framework provides the backend architecture, exposing a RESTful API that handles user inputs such as animal type, age, temperature, and selected symptoms, and returns a disease prediction generated by the machine learning model. On the frontend, **HTML, CSS, and JavaScript** are used to design a responsive and accessible user interface. Users interact with a form that dynamically responds to inputs, performs validation, and communicates with the Flask backend for inference. This creates a smooth and intuitive user experience.

	The full application is deployed on **Render.com**, which supports automatic builds from GitHub and seamless hosting of both backend services and static frontend files. Its simplicity and scalability make it well-suited for academic-level AI projects. Finally, **Power BI** is utilized as a complementary tool for data visualization and reporting. It allows the system to transform raw model predictions and dataset statistics into insightful dashboards, helping stakeholders such as veterinarians and livestock managers make informed decisions based on symptom patterns and disease distribution. **Rationale for Selection of Technologies:** **Python 3.8+:** Python is one of the most popular programming languages for machine learning and data science due to its simplicity, flexibility, and extensive ecosystem. It offers rich support for libraries such as Scikit-learn, Pandas, and NumPy, which accelerate the development of AI-powered applications. Its readability and compatibility with cloud platforms also make it ideal for academic and research-oriented projects. **Scikit-learn 1.3.2+:** Scikit-learn is a robust and user-friendly machine learning library that provides a wide range of classical algorithms. For this project, it enables fast experimentation with classification models, such as decision trees, and supports preprocessing techniques like label encoding and multi-label binarization. Its integration with Joblib also makes model deployment more manageable. **Flask 2.0+:** Flask is a lightweight web framework that is well-suited for serving machine learning models in a web environment. Its minimalistic structure allows rapid development of APIs, and it integrates smoothly with Python applications. In PredictVet, Flask is used to manage user input and route data between the frontend and the trained model. **HTML / CSS / JavaScript:** These core web technologies are used to build a clean and interactive user interface. They enable users to select animals, enter health parameters, and view prediction results seamlessly. Responsive design elements make the application accessible across different devices. **Joblib:** Joblib is used to serialize and compress the trained model, enabling fast loading during prediction without retraining. It is especially useful when deploying models on cloud platforms where storage space and speed are limited. **Render.com:** Render provides an accessible and developer-friendly cloud platform for deploying web applications. It supports GitHub-based deployment and is well-suited for Flask projects. Its free tier is ideal for academic and demonstration use cases. **Power BI:** Power BI adds a strong visual layer to the system. It transforms complex dataset statistics and model outputs into interactive dashboards, allowing stakeholders to track disease trends, symptom frequency, and potential risks. It enhances the interpretability and decision-making power of the system.

2. Key Implementation Aspects of the PredictVet System

2.1 System Architecture Overview

The PredictVet system is a small online website that combines the ability of machine learning with interactive form-based input. It works on a client-server basis, whereby a user-friendly frontend interface reachable via a browser interacts with a backend model created in Python. The main parts of the system are listed here briefly:

- A trained classification model (RandomForestClassifier)
- Preprocessing pipeline (encoding, binarization, validation)
- Flask-based backend API
- HTML/CSS/JS frontend interface
- Render.com deployment for live hosting
- Power BI for dataset visualization and model insight dashboards

The architecture is designed around a simplified MVC pattern:

- Model: Trained .joblib machine learning model
- View: HTML-based input interface with user interactivity
- Controller: Flask route logic handling inputs and predictions

2.2 Data Preprocessing and Dataset Construction

The dataset used in the model was not limited to the original open-access source but was significantly enhanced to better reflect real-world veterinary practices. Instead of using the Kaggle[1] dataset directly, ten additional diseases were manually added based on internationally recognized sources. The initial dataset for PredictVet was sourced from Kaggle's Livestock Disease Diagnosis Dataset (Captain_Gee, 2021). While the original dataset contained useful mappings of symptoms to diseases, it was significantly expanded with additional conditions and species-based references derived from veterinary literature to increase model reliability and applicability. These include Brucellosis[15] (CDC), Mastitis[16] (DairyNZ), Bovine Viral Diarrhea[17] (BVD) (Merck Vet Manual), Enterotoxemia[18] (Pulpy Kidney) (Oregon State Extension), Theileriosis[19] (FAO), Q-Fever[20] (CDC), Rabies[21] (WHO), Johne's Disease[22] (Merck Vet Manual), Mange[23] (NADIS), and Contagious Caprine Pleuropneumonia[24] – CCPP (FAO).

Key preprocessing steps included:

- Label Encoding for animal and disease names
- MultiLabelBinarizer for converting multi-symptom entries into binary arrays
- Validation checks for age and temperature ranges based on species-specific biological norms

```
from sklearn.preprocessing import LabelEncoder, MultiLabelBinarize

df = pd.read_csv("animal_disease_dataset_extended.csv")

le_animal = LabelEncoder()
df['animal_encoded'] = le_animal.fit_transform(df['Animal'])

le_disease = LabelEncoder()
df['disease_encoded'] = le_disease.fit_transform(df['Disease'])

symptom_cols = ['Symptom 1', 'Symptom 2', 'Symptom 3']
df['all_symptoms'] = df[symptom_cols].values.tolist()
mlb = MultiLabelBinarizer()
symptom_features = mlb.fit_transform(df['all_symptoms'])
```

Figure 1: Data Preprocessing and Dataset Construction

2.3 Model Training, Evaluation and Serialization

The classification model is derived from the scikit-learn `RandomForestClassifier`. Its robustness, great tabular data accuracy, and multiclass classification problem handling skills all provide justification for its adoption. As it is a collection of decision trees, the model has lower risk of overfitting than a single decision tree would have. Model-setting used 100 estimators (n_estimators=100) with a random seed (random_state=42) for regularity and reproducibility.

To assess the model's actual performance, the data set was divided into 80% training and 20% testing. Retraining on the training portion resulted in a test set **accuracy of 81%** for the model. Thus, performance across important disease groups is quite consistent: **a weighted average F1 score of 0.81**. With a **macro average F1 score of 0.63**, the model's fairly poor predictive ability for less common disease classifications is shown.

This constraint is mostly due to the data imbalance; certain less prevalent diseases had only a few cases, therefore the model had a very difficult time identifying their trends. This emphasizes the need of applying methods like class weighting or oversampling to balance into its assessment for improved future performance.

Using the growing package for serializing the model—along with its related label encoders—and storing them in a separate `/model` folder. They guarantee compatible training and production environments during inference.

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2032
1	1.00	1.00	1.00	2022
6	1.00	1.00	1.00	1902
8	0.40	0.39	0.40	1393
10	0.00	0.00	0.00	1
11	0.41	0.41	0.41	1408
accuracy			0.81	8758
macro avg	0.63	0.63	0.63	8758
weighted avg	0.81	0.81	0.81	8758

Figure 2: The model's performance's score

2.4 Flask Backend Implementation

The Flask backend handles routing, form data parsing, validation, and communication with the trained model. The /predict route handles user input, checks necessary fields, converts symptom choices into feature vectors, and sends them to the loaded model for prediction. If the user inputs data outside of specified ranges, or the user did not complete the necessary requirements, error messages are displayed.

```
@app.route('/predict', methods=['POST'])
def predict():
    animal = request.form.get('animal')
    age = request.form.get('age')
    temperature = request.form.get('temperature')
    symptoms = request.form.getlist('symptoms')

    animal_list = sorted(animal_info.keys())
    ref = animal_info.get(animal)
    selected = symptoms

    if not animal or not age or not temperature:
        return render_template('index.html', prediction_text='Please fill all fields.',
                               summary={
        "animal": animal,
        "age": age,
        "temperature": temperature,
        "symptoms": symptoms
    },
    animal_list=animal_list, animal=animal,
                               age=age, temperature=temperature,
                               symptoms_selected=selected, reference_info=ref,
                               symptom_groups=symptom_groups)
```

Figure 3: Flask Backend Implementation

2.5 Frontend Interface and User Flow

The frontend of the system is designed using HTML, CSS, and JavaScript. Users interact with the system by selecting an animal, entering the animal's age and temperature, and indicating one or more symptoms from a large number of checkbox categories. After clicking the predict button, the backend processes the inputted data, and displays the most likely illness associated with the symptom vectors. Usability improvements of the interface include input validation, auto-scrolling to the results section, and a summary box referring to user's selections.

2.6 Deployment via Render.com

The system is deployed on Render.com with:

- requirements.txt for dependency setup
- Procfile for server configuration
- GitHub-based CI/CD
- The system is publicly accessible via Render.com at the following URL:https://predictvet.onrender.com
- Github link: https://github.com/aysegulakyuuz/PredictVet.git

2.7 Visual Analytics with Power BI

Power BI was used to visualize key patterns in the dataset and support interpretation of model input. Created visuals include:

- A bar chart showing the frequency of symptoms
- A stacked column chart displaying diseases by animal type
- A line chart comparing average age and body temperature by disease

These visuals help explain the distribution of symptoms and conditions, and enhance the practical understanding of the model's scope.

3. Screenshots and Visualizations

3.1. Home Page – Input Form: This screen shows the main user interface where animal type, age, body temperature, and symptoms are entered. The interface uses grouped symptom categories and inline validation messages for enhanced usability.

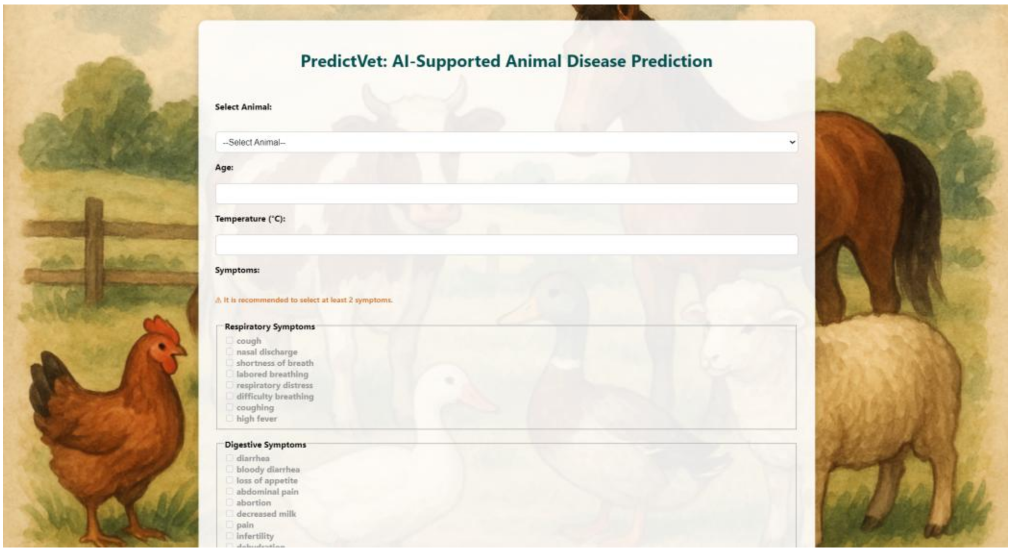


Figure 4: Home Page – Input Form

3.2. Input Validation Warning: If the user does not provide valid input (e.g., missing symptoms or unrealistic temperature), the system shows a specific warning message. This helps ensure data quality for prediction.

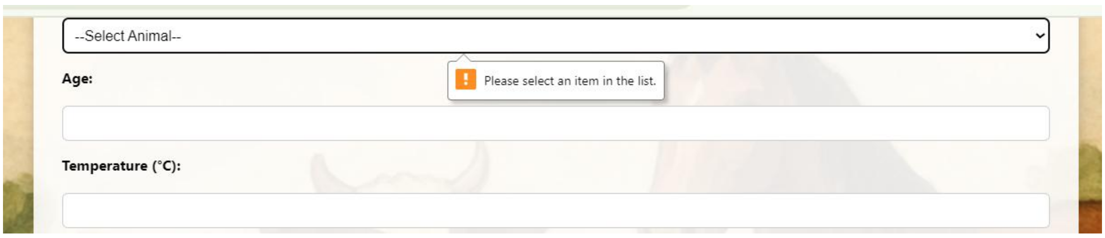
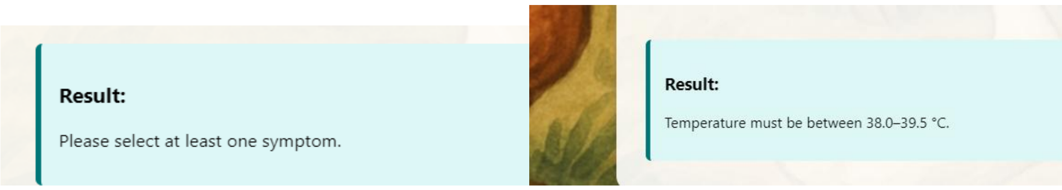


Figure 5: Input Validation Warning



3.3. Prediction Result View: After a valid form submission, the system predicts the most likely disease and displays it along with a summary of the entered data.

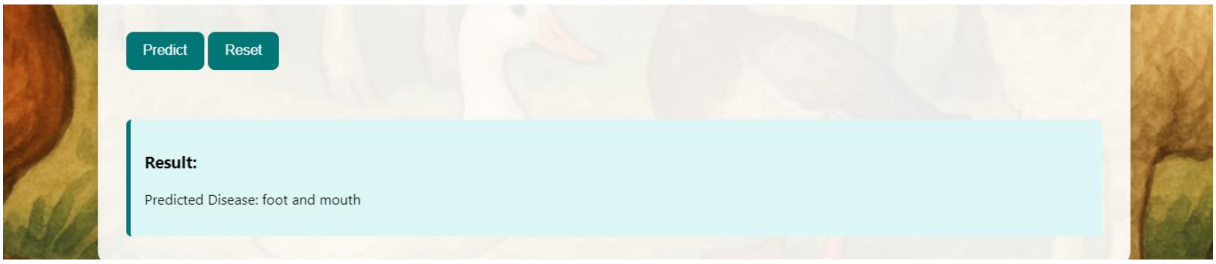


Figure 6:Prediction Result View

3.4. Power BI Symptom Frequency Chart: This chart shows how often each symptom appears in the dataset. "Loss of appetite", "depression", and "difficulty walking" are the most common.

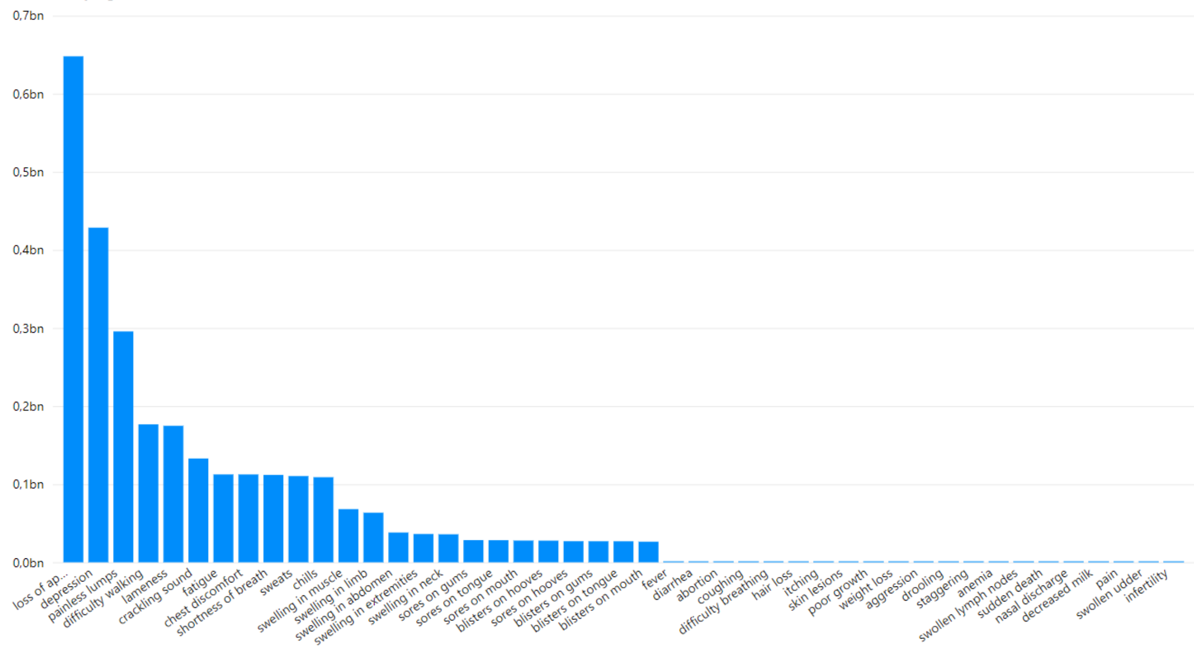


Figure 7: Power BI Symptom Frequency Chart

3.5. Power BI – Disease by Animal Type: This chart shows which diseases are most common for each animal type. It helps visualize species-specific disease patterns.

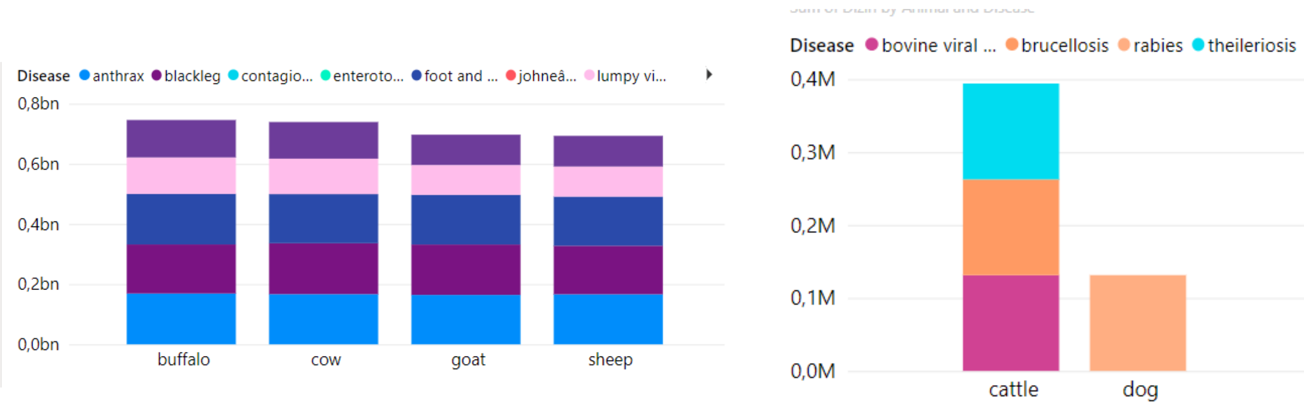


Figure 8: Disease by Animal Type

3.6. Power BI – Average Temperature and Age by Disease: This dual-line chart compares the average age and body temperature (°C) of animals diagnosed with different diseases. Temperature values were converted from Fahrenheit to Celsius where necessary.

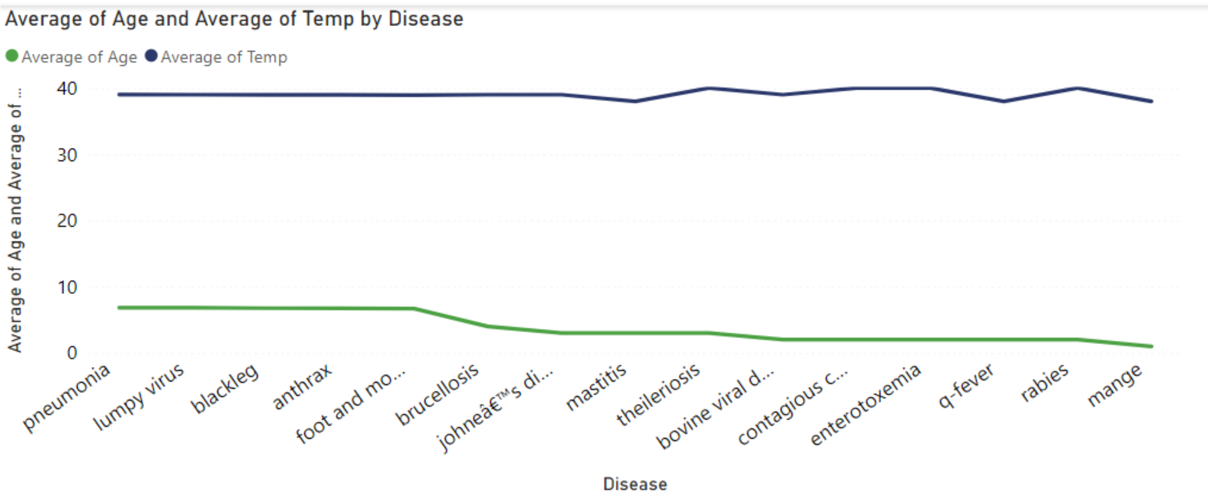


Figure 9:Average Temperature and Age by Disease

4. Conclusions and Development Prospects

The PredictVet project to deliver the capability to forecast animal illness derived from user inputs (e.g., species, age, temperature, and symptoms) using an AI powered web tool met her main goal. The solution was developed using Flaks and HTML/CSS/JavaScript and included a RandomForestClassifier model trained on the provided dataset along with an interesting front end. Render.com hosted the solution and allowed the app to be accessed in real-time by a successfully deployed application.

As applied, data preprocessing techniques such as label encoding, symptom binarization, and biological range validation added value to the dataset. In addition to open-source documentation, human-checked symptoms and disease information from veterinary sources (e.g., FAO, CDC, and scientific journals) was manually added to improve model accuracy and reliability. Power BI visualization was employed to analyze trends in symptom frequency, distribution of diseases by species, and correlation of age or temperature with specific diseases.

The model can make uniform predictions on common diseases, and the interface provides ease of use in the form of symptom picking groups, input validation, and easy-to-read results. These features enhance its usability for field applications so that veterinarians and livestock managers can make decisions rapidly.

Although the model is extremely good on adequately represented classes, limited prediction on uncommon diseases is still a limitation in the form of class imbalance. As demonstrated in experimental evaluations, the system achieved 81% accuracy and a weighted F1-score of 0.81, while the macro-averaged F1-score of 0.63 suggests scope for future improvement across underrepresented categories. In the future, PredictVet can be expanded by expanding the data base to more animals and diseases, including prediction confidence levels, allowing symptom input in terms of duration and severity, implementing multilingual support, and building a mobile app for offline use in remote locations.

In conclusion, PredictVet provides a robust and scalable foundation for AI-enabled veterinary diagnostics. It combines predictive accuracy with usability and effective data visualization, offering real-world practicality in animal husbandry and enabling further convergence and integration.

Bibliography

1. Kaggle. (n.d.). *Animal Disease Symptoms Dataset*. Retrieved from https://www.kaggle.com/datasets/captaingee/livestock-disease-diagnosis-dataset
2. Scottish Rural College (SRUC). (n.d.). *IntelliPig: A Face-Based Automated On-Farm Pig Health Monitoring System*. Retrieved from https://pure.sruc.ac.uk/en/projects/a-face-based-automated-on-farm-pig-health-monitoring-system-intel/fingerprints/
3. Performans Dergileri. (2023). *Artificial Intelligence Applications for Veterinarians*. Retrieved from https://performansdergileri.com/veteriner-hekimler-icin-yapay-zeka-uygulamalari-h
4. Webrazzi. (2025). *Using AI to Analyze Animals' Emotional States*. Retrieved from https://webrazzi.com/2025/02/17/arastirmacilar-yapay-zekayi-hayvanlarin-duygularini-yorumlamasi-icin-egitiyor
5. Ultralytics. (2023). *Monitoring Animal Behavior Using YOLOv8*. Retrieved from https://www.ultralytics.com/tr/blog/monitoring-animal-behavior-using-ultralytics-yolov8
6. Perfume Database. (n.d.). *Farm Sense: Behavioral and Scent Monitoring for Pigs*. Retrieved from https://www.perfumedatabase.com
7. Journal of Toxicological Sciences Retrieved from https://academic.oup.com/toxsci
8. FAO. (2020). *Animal Health Manual: Veterinary Epidemiology and Disease Control*. Food and Agriculture Organization of the United Nations. Retrieved from https://www.fao.org/animal-health/en/
9. CDC. (n.d.). *One Health Initiative*. Centers for Disease Control and Prevention. Retrieved from https://www.cdc.gov/one-health/about/index.html
10. Wiley Online Library. (n.d.). *Journal of Veterinary Microbiology and Immunology*. Retrieved from https://onlinelibrary.wiley.com/journal/13652052
11. Wageningen University & Research. (n.d.). *Animal Sciences and AgriTech Research*. Retrieved from https://www.wur.nl/en.htm
12. Merck Veterinary Manual. (n.d.). *Common Livestock Diseases*. Retrieved from https://www.merckvetmanual.com
13. Render.com. (n.d.). *Web App Deployment Documentation*. Retrieved from https://render.com
14. Power BI. (n.d.). *Microsoft Power BI Documentation*. Retrieved from https://learn.microsoft.com/en-us/power-bi/
15. Brucellosis Retrieved from https://www.cdc.gov/brucellosis/index.html
16. Mastitis. Retrieved from https://www.dairynz.co.nz/animal/cow-health/mastitis/
17. Bovine Viral Diarrhea (BVD). Retrieved from https://www.merckvetmanual.com/digestive-system/diseases-of-the-stomach-and-intestines-in-ruminants/bovine-viral-diarrhea-bvd-in-cattle
18. Enterotoxemia (Pulpy Kidney). Retrieved from https://extension.oregonstate.edu/enterotoxemia-sheep-goats
19. Theileriosis. Retrieved from https://www.fao.org/theileriosis/en/
20. Q-Fever. Retrieved from https://www.cdc.gov/qfever/index.html
21. Rabies (Kuduz). Retrieved from https://www.who.int/news-room/fact-sheets/detail/rabies
22. Johne's Disease. Retrieved from https://www.merckvetmanual.com/digestive-system/paratuberculosis-johne-s-disease/paratuberculosis-johne-s-disease-in-ruminants
23. Mange (Uyuz). Retrieved from https://www.nadis.org.uk/disease-a-z/farm-animals/mange/
24. Contagious Caprine Pleuropneumonia (CCPP). Retrieved from https://www.fao.org/3/y0820e/y0820e00.htm