Project Design Phase-II

Solution Architecture

Date	22 October 2023
Team ID	PNT2022TMID592078
Project Name	Predicting Lumpy Skin Disease
Maximum Marks	4 Marks

Solution Architecture:

Data Collection and Preprocessing

1. Data Collection: Gathering historical lumpy skin disease cases from a reliable source, such as Kaggle. Ensured the data is relevant, accurate, and up-to-date.
2. Data Cleaning: Cleanimg and pre-process the data to remove inconsistencies, missing values, and outliers. Handle missing values appropriately, either by imputation or deletion.
3. Feature Engineering: Extracted the relevant features from the data. This may involve transforming categorical variables into numerical representations, creating new features from existing ones, and selecting the most informative features.
4. Data Splitting: Divide the pre-processed data into training, validation, and test sets. The training set is used to build the machine learning model, the validation set is used to tune the model's hyperparameters, and the test set is used to evaluate the model's performance on unseen data.

Machine Learning Model BUilding

1. Model Selection: We considered the algorithms like logistic regression, decision tree, gradient boosting trees like XGBoost.

2. Model Training: Train the model on the training set. This involves optimizing the model's parameters to minimize the classification error. ”train_test_split”,function from scikit-learn library that splits the data into training and testing sets. Here, we did a 4:1 split (80:20 i.e., training set is 80, whereas testing set is 20).
3. Hyperparameter Tuning:
   As, the accuracy is up to 97%, we decided to skip cross-validation.

4. Model Evaluation: Evaluating the performance of the trained model on the test set. This involves calculating metrics like accuracy, precision, recall, and F1-score to assess the model's ability to correctly classify lumpy skin disease cases.

Deployment and Monitoring

1. Model Deployment: Deploy the trained machine learning model in AWS. This may involve integrating the model into a web application and creating a Flask API.
2. Model Monitoring: Continuously monitor the performance of the deployed model in production. This involves tracking key metrics, such as accuracy and error rates, and detecting any performance degradation.

We can consider 'Model Retraining' periodically for better results.

Solution Architecture Diagram: