**RAINFALL PREDICTION USING MACHINE LEARNING**

AIMS:

Predicting rainfall is a challenging task, often even surpassing the accuracy of meteorological departments. In this article, we'll build a machine-learning model to predict rainfall using Python.

LIBRARIES:

1. **Pandas**: Data manipulation and analysis.

2. **Numpy**: Efficient numerical computations.

3. **Matplotlib/Seaborn**: Visualization.

4. **Sklearn**: Preprocessing, model training, and evaluation.

5. **XGBoost**: Gradient boosting for high accuracy.

6. **Imblearn**: Handling imbalanced datasets.

APPROACHES:

• Data Loading and Preprocessing

**Dataset**: Rainfall.csv.

**Size**: (366, 12).

Removed unnecessary spaces in column names.

Handled null values by imputing with column means.

Imbalanced data handled using **RandomOverSampler**

• **Exploratory Data Analysis**

1. Rainy days have lower max temperature, higher humidity, and less sunshine.

2. Windspeed increases on rainy days.

Highly correlated features (maxtemp & mintemp) were removed

• **Model Training:**

Logistic Regression

XGBClassifier

Support Vector Classifier (SVC)

• **Output**:

**Logistic Regression**: Training: 0.889, Validation: 0.896

**XGBClassifier**: Training: 0.990, Validation: 0.840

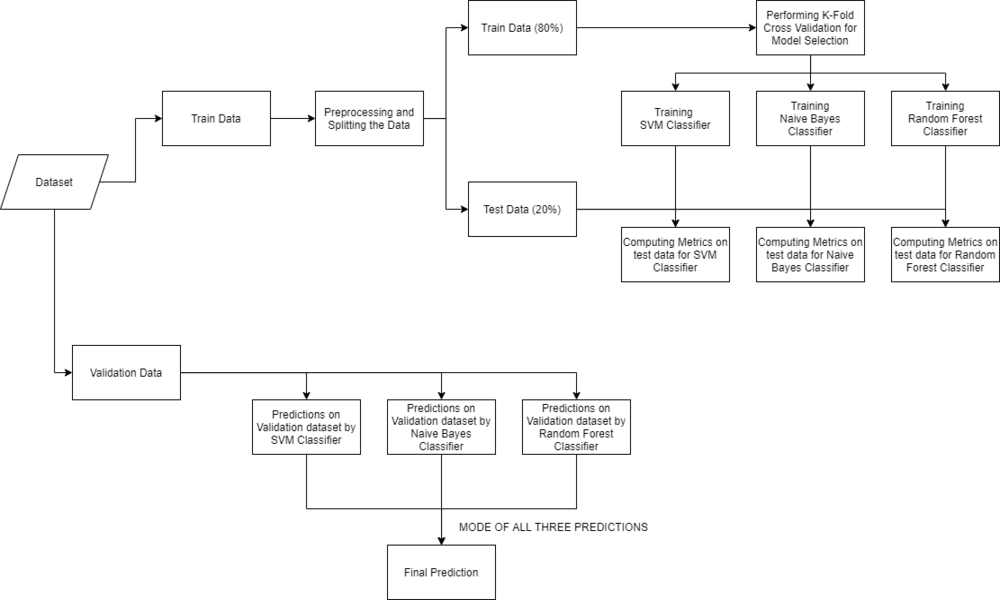
**SVC**: Training: 0.902, Validation: 0.885

• **Model Evaluation**

**Best Models**: Logistic Regression and SVC.

**Confusion Matrix** and **Classification Report** (using SVC):

This approach shows that **Logistic Regression** and **SVC** are effective for this problem, with well-balanced training and validation accuracies.

IMPLEMENTATION: 

READING THE DATASET:

Firstly we will be loading the dataset from the folders using the pandas library. While reading the dataset we will be dropping the null column. This dataset is a clean dataset with no null values and all the features consist of 0’s and 1s. Whenever we are solving a classification task it is necessary to check whether our target column is balanced or not. We will be using a bar plot, to check whether the dataset is balanced or not.



From the above plot, we can observe that the dataset is a balanced dataset i.e. there are exactly 120 samples for each disease, and no further balancing is required. We can notice that our target column i.e. prognosis column is of object datatype, this format is not suitable to train a machine learning model. So, we will be using a label encoder to convert the prognosis column to the numerical datatype. Label Encoder converts the labels into numerical form by assigning a unique index to the labels. If the total number of labels is n, then the numbers assigned to each label will be between 0 to n-1.

SPLITTING THE DATA FOR TRAINING & TESTING THE MODEL:

Now that we have cleaned our data by removing the Null values and converting the labels to numerical format, It’s time to split the data to train and test the model. We will be splitting the data into 80:20 format i.e. 80% of the dataset will be used for training the model and 20% of the data will be used to evaluate the performance of the models

MODEL BUILDING:

After splitting the data, we will be now working on the modeling part. We will be using K-Fold cross-validation to evaluate the machine-learning models. We will be using Support Vector Classifier, Gaussian Naive Bayes Classifier, and Random Forest Classifier for cross-validation. Before moving into the implementation part let us get familiar with k-fold cross-validation and the machine learning models.

* **K-Fold Cross-Validation:**K-Fold cross-validation is one of the cross-validation techniques in which the whole dataset is split into k number of subsets, also known as folds, then training of the model is performed on the k-1 subsets and the remaining one subset is used to evaluate the model performance.
* **Support Vector Classifier:**Support Vector Classifier is a discriminative classifier i.e. when given a labeled training data, the algorithm tries to find an optimal hyperplane that accurately separates the samples into different categories in hyperspace.
* **Gaussian Naive Bayes Classifier:**It is a probabilistic machine learning algorithm that internally uses Bayes Theorem to classify the data points.
* **Random Forest Classifier:**Random Forest is an ensemble learning-based supervised machine learning classification algorithm that internally uses multiple decision trees to make the classification. In a random forest classifier, all the internal decision trees are weak learners, and the outputs of these weak decision trees are combined i.e. mode of all the predictions is as the final prediction.

**Building robust classifier by combining all models:**

**Fitting the model on whole data and validating on the Test dataset:**

**Creating a function that can take symptoms as input and generate predictions for disease**

***Note: The symptoms that are given as input to the function should be exactly the same among the 132 symptoms in the dataset****.*