**A PROJECT REPORT**

**ON**

**Weather Prediction and Forecasting using Machine Learning Algorithms**

Submitted in partial fulfilment for the requirement of the award of

TRAINING

IN

Data Analytics, Machine Learning and AI using Python



*Submitted By*

**Neelesh T Rao (**R. V. College of Engineering, Bangalore**)**

*Under the guidance of*

**Mr. Ashish Saini**

**ACKNOWLEDGEMENT**

I would like to express sincere gratitude towards my guide Mr. Ashish Saini. He was a fabulous teacher, and covered all concepts related to the practical understanding of ML with great clarity. He was always ready to clear doubts of students, and suggested excellent resources to understand the subject further. I would also like to commend my batchmates for asking thought provoking questions and providing alternate solutions to problems. This further deepened my understanding of the subject. Finally, I would like to think my family for their unwavering support. Their ideas and solutions helped me improve my final project by leaps and bounds. I also declare to the best of my knowledge and belief that the Project Work has not been submitted anywhere else.

**Abstract**

Weather Forecasting and Prediction is a very important part of our daily lives. We plan our day-to-day activities based on external weather conditions. Accurate Weather Forecasting will no doubt improve the quality of our lives. Machine Learning has been used in this project to prepare a Weather Forecaster based on previous data with high accuracy. Random Forest Algorithm has been used to predict whether rain can be expected tomorrow or not. Even in real life weather forecasting, Random Forest Classifier is preferred because it provides for multiple trees with more branching. The dataset has been obtained from Kaggle, and has undergone sever preprocessing by filling null values, removing columns with too few entries, and converting all object datatype columns to integers by LabelEncoders. After splitting into training and testing samples, the default model provided by sklearn was trained and fitted. Next, Hyperparameter Tuning was done with the help of validation curve fitting. The best possible values of hyperparameters were chosen, and the model was again trained to obtain a reasonably higher accuracy. The model was then saved using pickle module, so that it could be loaded separately and used without having to retrain every time.

**Introduction to Machine Learning**

Machine Learning (ML) is a branch of AI that enables computers to learn from data and improve over time without being explicitly programmed. They are trained to identify patterns in data and learn from them in order to make their own predictions.

The two types of Machine Learning are:

1. Supervised Learning: These models are based on labelled training data. Each training sample includes an input and a desired output. The two main types of supervised learning are regression and classification. In classification, the output is a category with a finite number of options. In regression, the expected output is a continuous number. Hence, regression is used to predict quantities like the probability of an event, while classification is used to classify into categories.
2. Unsupervised Learning: It covers insights and relationships in unlabeled data. Models are fed input data but the desired outcomes are unknown, so inferences are made based on circumstantial evidence without guidance or training. Clustering is a type of unsupervised learning, which consists of grouping similar data. It is used to identify hidden patterns and trends.

**Problem Statement**

The task is to build an ML model to predict weather data using data of ten previous years. We must find the value of the Column RainTomorrow given other parameters like RainToday, temperature, air pressure and wind direction. The dataset is to be obtained from Kaggle and should have a minimum of 10 features.

**Methodology**

**Choice of Model**

For the task of Weather Prediction, it is evident that a **classification algorithm** must be used. However, care must be taken while choosing the algorithm to ensure best accuracy.

The possible options are:

1. K Nearest Neighbors

2. SVM

3. Decision Trees

4. Naïve Bayes

5. Random Forest

6. Logistic Regression

Upon research, it was found that **Random Forest** is the best possible option for Weather Prediction. It generates a tree like structure with traversal happening based on conditions, which is crucial in forecasting.

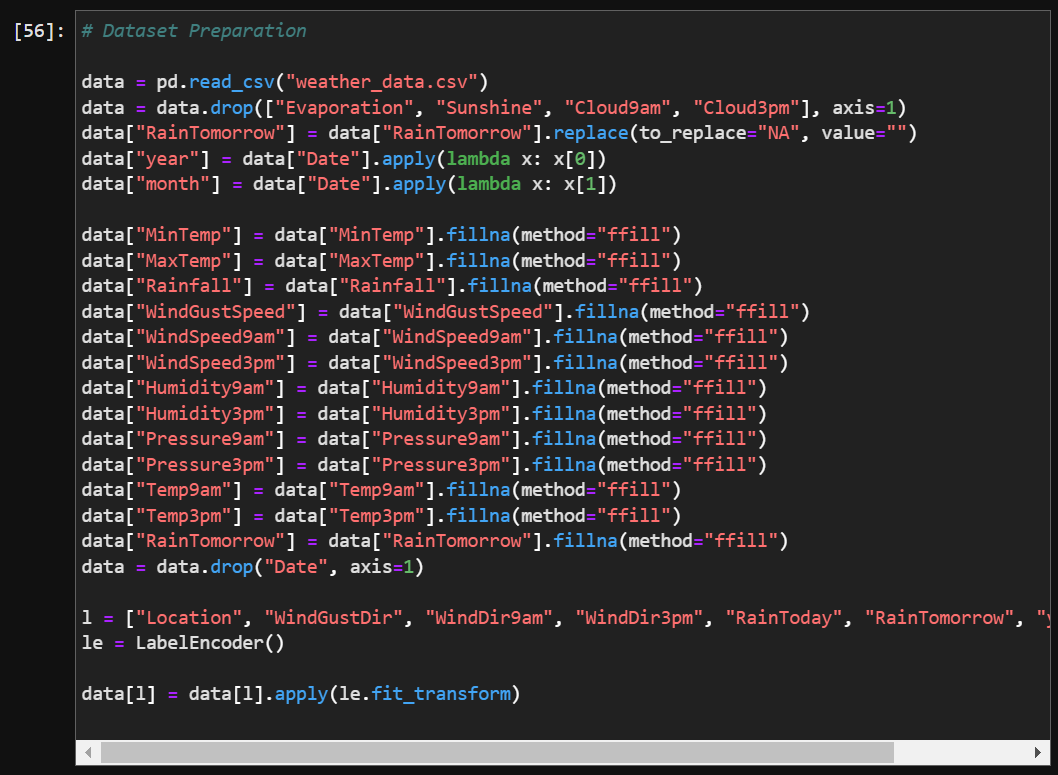
**Dataset and Preprocessing**

The next step was acquisition of a dataset. This was obtained from Kaggle website with the link: <https://www.kaggle.com/jsphyg/weather-dataset-rattle-package>

The dataset obtained here unfortunately was not clean. So, **preprocessing** had to be applied before it could be used for training the model.

It was observed that the columns Evaporation, Sunshine, Cloud9am and Cloud3pm had very few entries compared to the remaining columns. Hence, instead of filling them with random values, it was better to **drop** them from the dataset.

Next, null values for all the columns were filled using **fillna** (method=” ffill”), to copy the value of the previous entry into the null value. The thought process for this procedure is quite simple, as in most scenarios, today’s weather is heavily influenced by yesterday’s weather.

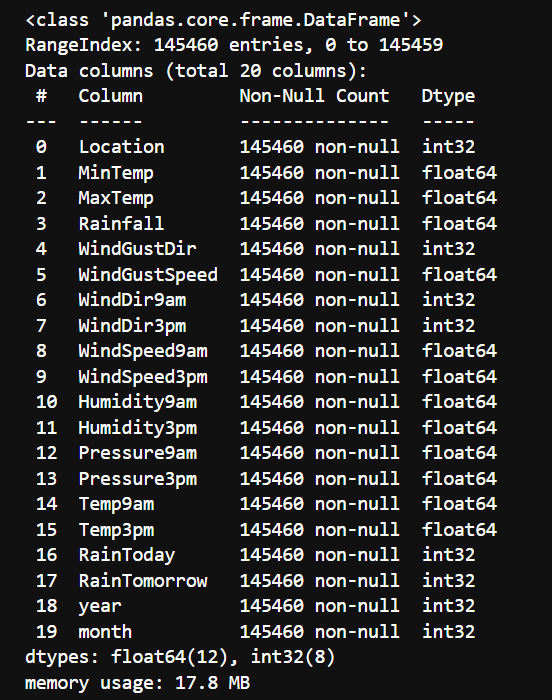


**Figure 1 – Code for Data Preprocessing**

Next, the date column was transformed into year and month by converting the string into a date object and accessing the year and month part of it. After this, the original date column was also dropped.

Finally, the **LabelEncoder()** from sklearn.preprocessing was used to give numeric values to all possible object classes, as only numeric values would be accepted into the model.

Finally, printing the info of the data gives the following output:



**Figure 2 – Preprocessed data**

We have now obtained **19 columns** with no null values, and all the types are numeric. Thus, preprocessing was successfully done to obtain a clean and proper dataset.

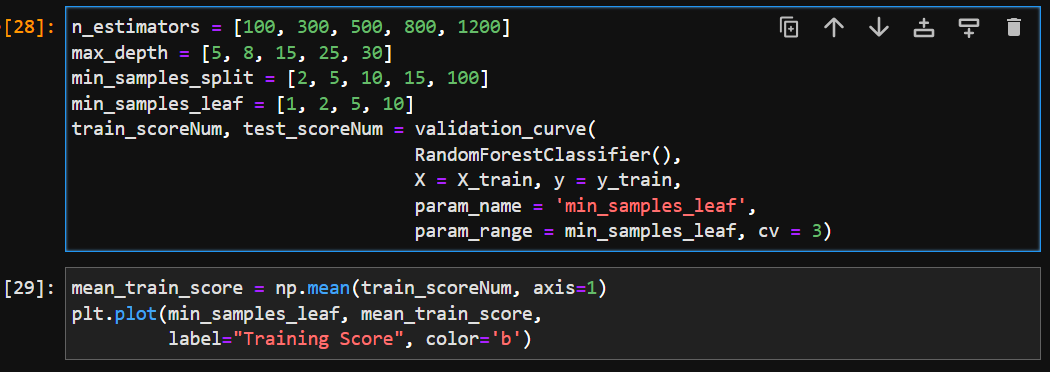
**Training and Testing Dataset**

The next step is splitting the dataset obtained above into the training and testing dataset. The training part is used to train the model, while the testing part is used to check the model’s accuracy.

Using the **train\_test\_split** function of the sklearn.model\_selection library, the dataset was split with 23% of the samples being assigned as testing samples.

**HyperParameter Tuning**

Hyper parameters are those parameters whose values control the learning process and determine the values of model parameters that a model learns. The process of choosing the right parameters and their best values is called hyper parameter tuning. These should not be confused with normal parameters like weights and biases. Instead, these are hyper parameters, which are specified for the model to be used.



**Figure 3 – Code for Hyper Parameter Optimization**

For the Random Forest Classifier, the important hyper parameters are:

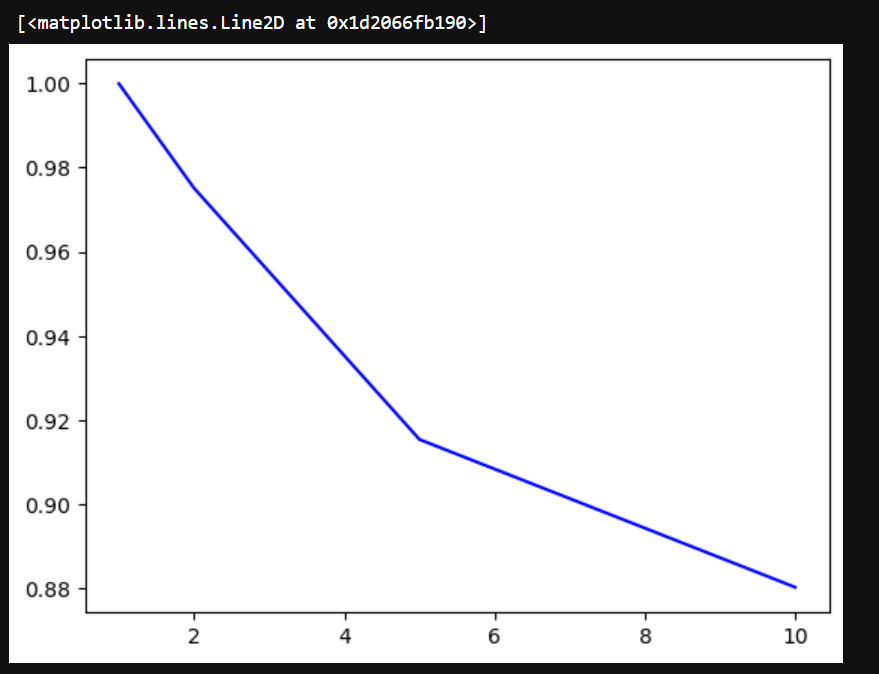
1. **n\_estimators**: It represents the number of trees in the forest. Usually, a higher number of trees allows for better learning of data. However, too many trees slow down the training and might lead to over fitting. Hence, hyper parameter optimization gives the best possible value.

2. **max\_depth**: It is the number of splits that each decision tree is allowed to make. If the number of splits is too low, the model is under fitted and if it is too high, the model is over fitted. Hence, hyper parameter optimization helps use choose the best value for max\_depth.

3. **min\_samples\_split**: It specifies the minimum number of samples an internal node must hold in order to split into further nodes. If the value is too low, the trees grow in height and overfits the data.

4. **min\_samples\_leaf**: It specifies the minimum number of samples to be present at the leaf nodes.

**Validation Curves** were used to plot the model performance metrics of accuracy against a range of hyper parameter values. For the current model we can observe that between 1 and 2, the highest accuracy is obtained for the min\_samples\_leaf hyper parameter. Hence, this is the value to be used during training.

****

**Figure 4 – The validation curve for min\_samples\_leaf**

**RandomForestClassifier**

In Random Forest Classification, **multiple decision trees** are created using different random subsets of data and features. Each decision tree provides an opinion on how to classify data. Predictions are made by taking the predictions of each decision tree and finding out the most popular result.

****

**Figure 5 – Random Forest Classification Image**

Sklearn toolkit provides a **RandomForestClassifier()** model, which can be imported from sklearn.ensemble.

The model must be initialized with the hyperparameters obtained from Hyper Parameter Optimization. They are:

1. n\_estimators = 800

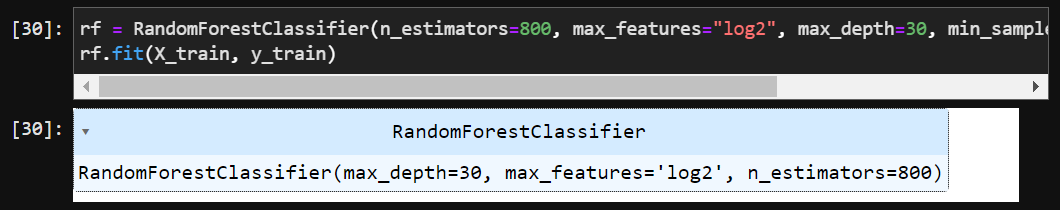
2. max\_features = log2

3. max\_depth = 30

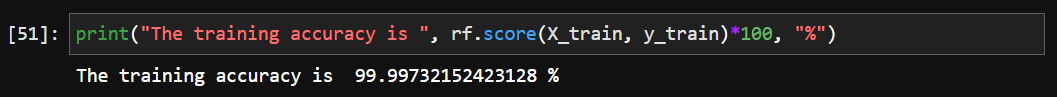
4. min\_samples\_split = 2

5. min\_samples\_leaf = 1

Now we must fit the training data, so that the model can learn the different features.

****

**Figure 6 – Training of RandomForestClassifier**

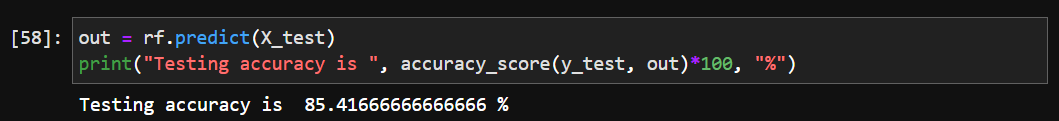
****

**Figure 7 – Training Accuracy**

**Results and Inferences**

**Testing dataset and Accuracy**

Now that we have trained the model, it is time to use the testing dataset to predict its accuracy. Important point to be remembered is that the model will have no idea of the testing dataset, and will be predicting the output freshly.

****

**Figure 8 – Testing Accuracy**

We can see that we obtained a reasonable accuracy of 84.415%. This value is higher than the value obtained without using any parameters by 2%.

**Saving the model**

The model cannot be retrained every time that it must be used, as it is a waste of time and computational power. So, we must save this model so that it can be loaded and used as and when we require.

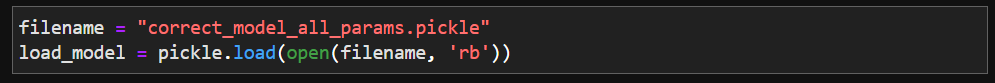
The pickle model can be used to save a model and reload it later.

****

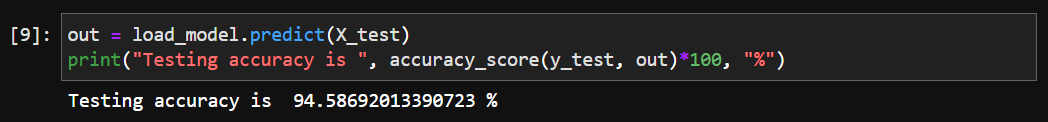
**Figure 9 – Saving the model as a binary file**

**Loading the Model**

The saved model can be loaded again using pickle for reuse.

****

**Figure 10 – Loading the model from a binary file**

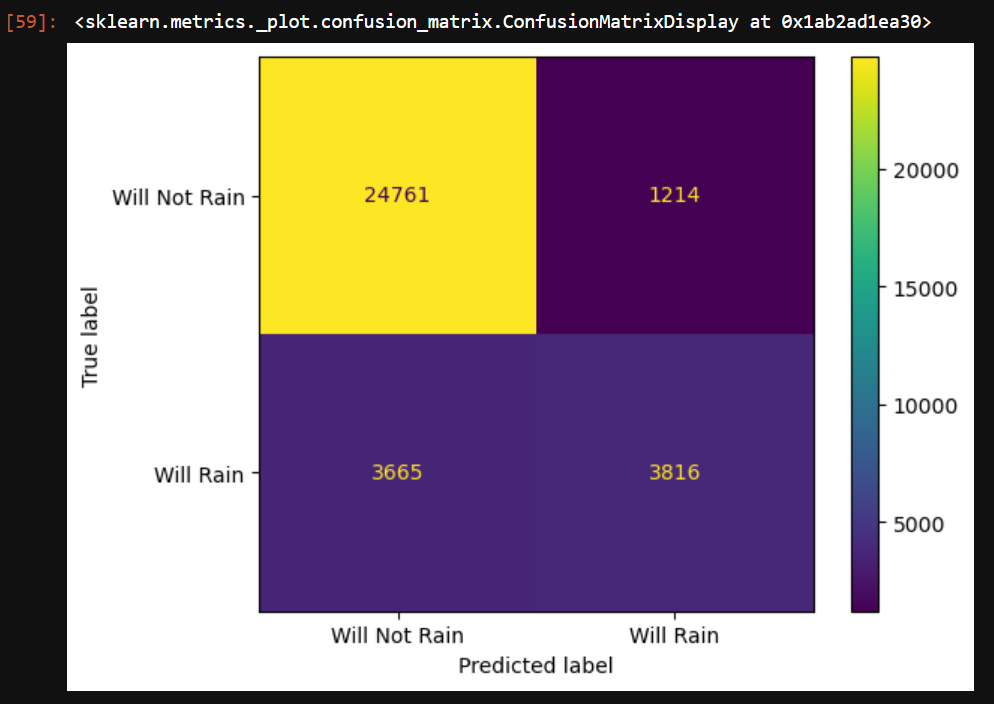
****

**Figure 11 – Using the loaded model for further prediction**

We have obtained a testing accuracy of 96.586% for this data.

**Confusion Matrix**

A Confusion Matrix is a diagram to get a good understanding of how well the classification has been performed.

****

**Figure 12 – Confusion Matrix**

Here, we can see the True Labal and the predicted Label. The main diagonal shows the correct classification, while the non-main diagonal shows the misclassification.

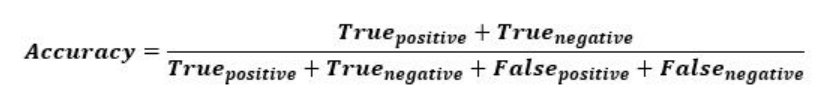
TP (Will Not Rain Correctly Classified) = 24761

TN (Will Rain Correctly Classified) = 3816

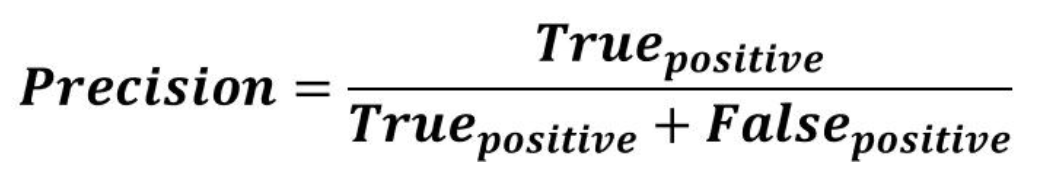
FN (False Negative) = 1214

FP (False Positive) = 3665

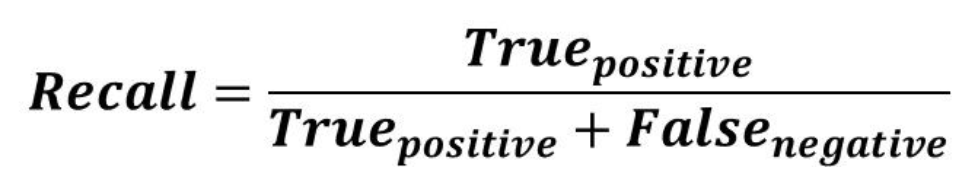
**Accuracy Metrices**

****

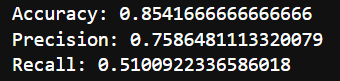
Accuracy: It speaks of how well the model performs across all classes.



Precision: It is the ratio of the number of positive samples correctly classified to the total number of samples classified as Positive (correct or incorrect).



Recall: It is the ratio of the number of positive samples correctly classified as positive to the total number of positive samples.

****

**Figure 13 – Accuracy Metrics Output**

**Conclusion**

In this project, a Random Forest Classifier was successfully implemented and tested. The obtained data was first cleaned and preprocessed. Next, the model was chosen along with the necessary hyper parameters. To improve model accuracy, Hyper Parameter optimization was also performed. Following this, the model was trained twice, once on default values and again with the new hyper parameters. These models were saved as binary files to allow for reuse.

Finally, the predictions were made and model accuracy was evaluated using Confusion Matrix and other accuracy metrices like precision and recall.

**Bibliography**

<https://www.kaggle.com/jsphyg/weather-dataset-rattle-package>

<https://blog.paperspace.com/deep-learning-metrics-precision-recall-accuracy/#:~:text=Accuracy%20is%20a%20metric%20that,the%20total%20number%20of%20predictions>.

<https://www.geeksforgeeks.org/validation-curve/>

<https://www.datacamp.com/tutorial/random-forests-classifier-python>

<https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac>