Rainfall Prediction using Feature Engineering Techniques

P. Satish,I.Harshith and V. Madhusudhan| Dr. Karthikeyan M| SENSE

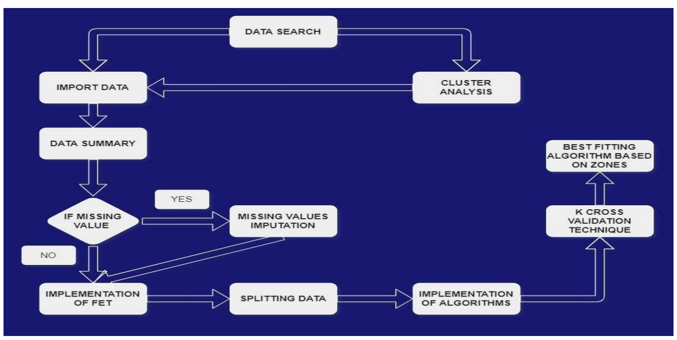
Motivation/ Introduction

Rainfall Prediction Systems are one of the fastest and most reliable procedures to predict rainfall distributions and screen floods. The systems use Internet of Things (IoT) devices to obtain data, and combine this data with Machine Learning and Data Engineering Techniques for the estimation of the amount of rainfall. Our primary objective in this work is to design and demonstrate a highly effective rainfall prediction system, and compare it to the existing works to show an improvement in its reliability and efficiency.

SCOPE of the Project

The target of our work is to use various Data Preprocessing and Feature Engineering techniques to transform and shape the data collected from atmospheric factors such as temperature, atmospheric pressure, wind speed, etc., and feed the processed data into three Machine Learning algorithms - Linear Regression, Random Forest, and XGBoost, each of which predict the occurrence of rainfall in a specific area. Then, we derive a comparison of the performance of each algorithm, and arrive at a conclusion of which algorithm is best suited and most reliable for the prediction. Finally, we perform an analysis of various clusters in the area using k-means clustering algorithm, to determine the best suited prediction algorithm for each cluster, helping us to determine the best prediction technique overall with higher certainty.

Methodology



→ The raw data is collected from the dataset "Rain in Australia" from kaggle. The dataset is imported into jupyter notebooks along with a collection of various python libraries used for data analysis and machine learning. Then, the following Feature Engineering techniques are used to shape the data accordingly.

→ <u>Feature Engineering Techniques</u>:

The feature engineering techniques implemented are described as follows:

- *Imputation* - Imputation is used when there are any missing values or data present in the dataset, and they can be filled in without needing to remove and reduce the size of the dataset.

  Mean/median missing value imputation replaces the missing spaces with the calculated mean/median of the non-missing values in that category.

- *Data Normalization* - Normalization is the process of scaling the data in the dataset to fit the values into a smaller range. It is useful when different attributes in the data are in different scales, which affects the overall efficiency of the algorithm, especially when the scales are significantly high in difference. Normalization fits all the sata to one simple scale, such as (0 to 1), or (-1 to 1).

- *Label Encoding* - The various labels of different variables in the data are converted from words to unique numbers, to convert them into a more machine-readable form. The categories are usually numbered 0,1,2,etc.

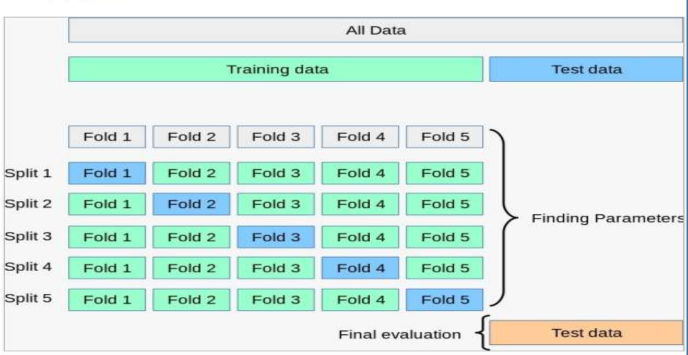


Figure 2 - Working of K-Cross Validation technique

Results

- First, we train the *Linear Regression Algorithm*:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
cv=KFold(n_splits=5,random_state=1,shuffle=True)
model = LinearRegression()
scores = cross_val_score(model, X1, y1, scoring='neg_mean_absolute_error',
                         cv=cv, n_jobs=-1)
print('MAE: ',np.mean(np.absolute(scores)))
scores = cross_val_score(model, X1, y1, scoring='neg_mean_squared_error',
                         cv=cv, n_jobs=-1)
print('RMSE: ',np.sqrt(np.mean(np.absolute(scores))))
```

```
MAE:  2.5049436124630313
RMSE:  7.176155438501579
```

- Next, we train the *Random Forest Algorithm*:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
cv=KFold(n_splits=5,random_state=1,shuffle=True)
model=RandomForestRegressor(n_estimators=30)
scores = cross_val_score(model, X1, y1, scoring='neg_mean_absolute_error',
                         cv=cv, n_jobs=-1)
print('MAE: ',np.mean(np.absolute(scores)))
scores = cross_val_score(model, X1, y1, scoring='neg_mean_squared_error',
                         cv=cv, n_jobs=-1)
print('RMSE: ',np.sqrt(np.mean(np.absolute(scores))))
```

```
MAE:  1.9200614256000499
RMSE:  6.644454074966145
```

- Then, the *XGBoost Algorithm*:

```
from xgboost import XGBRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
cv=KFold(n_splits=5,random_state=1,shuffle=True)
model=XGBRegressor(n_estimators=25)
scores = cross_val_score(model, X1, y1, scoring='neg_mean_absolute_error',
                         cv=cv, n_jobs=-1)
print('MAE: ',np.mean(np.absolute(scores)))
scores = cross_val_score(model, X1, y1, scoring='neg_mean_squared_error',
                         cv=cv, n_jobs=-1)
print('RMSE: ',np.sqrt(np.mean(np.absolute(scores))))
```

```
MAE:  1.8381865266062394
RMSE:  6.418036020025113
```

B. <u>Performance Comparison with Base Reference Paper:</u>

➢ We display below the values obtained from each of the three algorithms in our base reference paper, and a comparison of them with the values obtained in our work:

Table 4 - Performance comparison of our design vs.base reference paper

	MAE		RMSE	
Algorithm	Base Paper	Our Design	Base Paper	Our Design
Linear Regression	4.97	2.485	8.61	7.239
Random Forest	4.49	1.872	8.82	6.589
XGBoost	3.58	1.787	7.85	6.393

Conclusion/ Summary

Our design for the prediction of rainfall has provided optimal results under most circumstances. It has shown great improvement in accuracy scores in comparison to our reference paper. The analysis of clusters has also shown how our algorithm can perform when provided with different types of data in different distributions. It has proven to be more time efficient, highly reliable, and simple to understand and execute, and is therefore ready to be used in real time to predict rainfall in a location

Contact Details

iharshith568@gmail.com vmadhusudhan0951@gmail.com
psatish.sarma2018@vitstudent.ac.in

Acknowledgments/ References

1. Liyew, Chalachew & Melese, Haileyesus. (2021). Machine learning techniques to predict daily rainfall amount. Journal of Big Data. 8. 10.1186/s40537-021-00545-4.
2. Atitallah, S.B., et al.: Leveraging Deep Learning and IoT big data analytics to support the smart cities development: review and future directions. Comput. Sci. Rev. 38, 100303 (2020)
3. Dai, Weijun & Tang, Yanni & Zhang, Zeyu & Cai, Zhiming. (2021). Ensemble Learning Technology for Coastal Flood Forecasting in Internet-of-Things-Enabled Smart City. International Journal of Computational Intelligence Systems.14. 166. 10.1007/s44196-021-00023-y.