Rainfall Prediction

Telecom Sud Paris Internship Report

Author – Ayush Tankha
Email – ayush.tankha@essec.edu

1.INTRODUCTION

1.1 Mentors

This project in undertaken under the guidance of Prof. Anis Laouiti and Dr. Asma Lahbib in in the R2M (Réseaux et Services Multimedia Mobiles) department.

1.2 Problem Statement

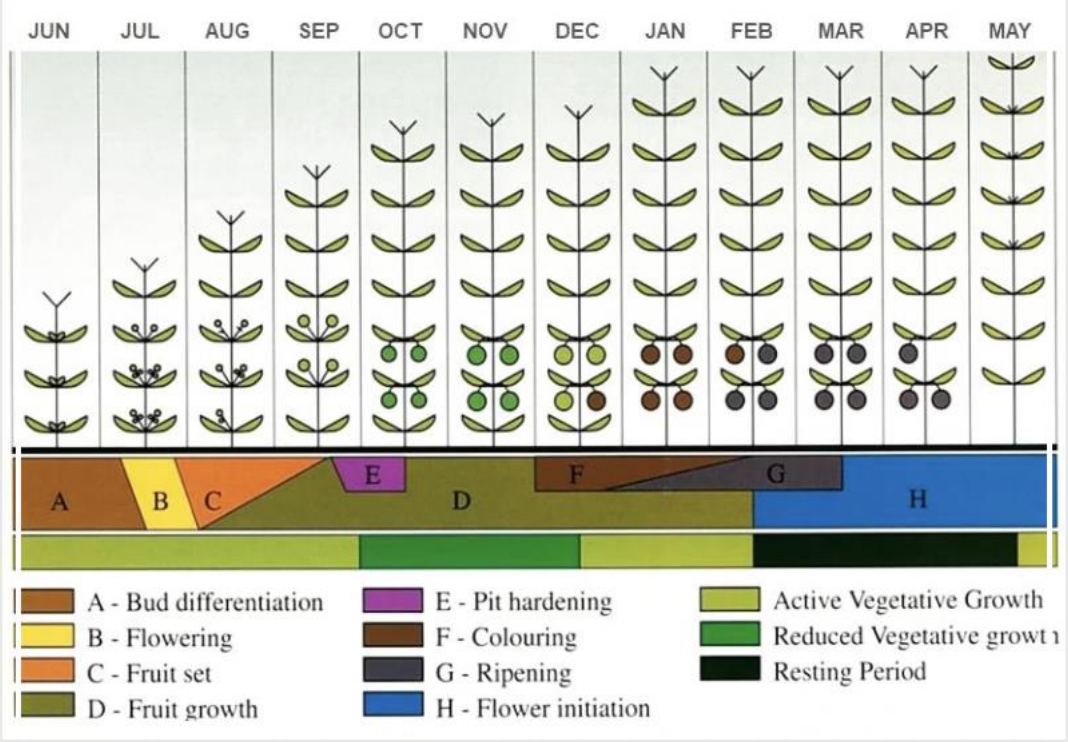
Tunisia is one of the largest exporters and producer of Olive Crops in the entire world. Farmer often face unpredictability of climatic conditions due to climate change. This project tries to cover a small segment of rainfall prediction meter for farmers. The overall project is still ongoing and plans to build an intermediary platform for farmers and customers of Tunisia.

1.3 Executive Summary

Tunisia > Spain > Italy hold the majority share of export market of Olive oil.[1] (Tunisia being 3rd largest in the world). Numerous studies have pointed out the importance of olive yield forecasting of airborne pollen amounts and weather conditions during the months following olive pollination.

Olive trees are influenced by human factors (e.g., agronomic techniques), environmental conditions (water deficit, extreme temperature) and phyto-pathological problems during both the pre-flowering and post-pollination periods. Experimentation and Methodology involved - Identifying Olive cultivation areas, Olive Flowering Monitoring, Climate Change Diagnosis, Statistical Analysis and Future Projections.[2]

The olive pollen production shows a variation related to temperature and rainfall during the dormancy period. A correlation was found between the number of olive tree inflorescences and airborne pollen counts.[3]. On the other hand, it also examines the vulnerability of olive groves to climate change.



Growth Cycle of Olive Crops [4]

2. MATERIAL & METHODS

2.1 Objective of Report

The objective of the report to present a thorough analysis and procedure behind building a rainfall predictor for the farmers of Tunisia. The report covers methods of data collection, data cleaning, data processing and building machine learning models for accurate prediction.

2.2 Data Collection

The data is collected from NASA Data Access Viewer for the city Sidi Bouzid with coordinates – 35.0354° N, 9.4839° E. The data was collected from 01/01/1983 through 08/01/2023.

The data was collected in tabular format (CSV) and had the following features-

Parameter(s):

- All Sky Surface Shortwave Downward Irradiance (kW-hr/m^2/day)
- All Sky Insolation Clearness Index (dimensionless)
- All Sky Surface Longwave Downward Irradiance (W/m^2)
- All Sky Surface PAR Total (W/m^2)

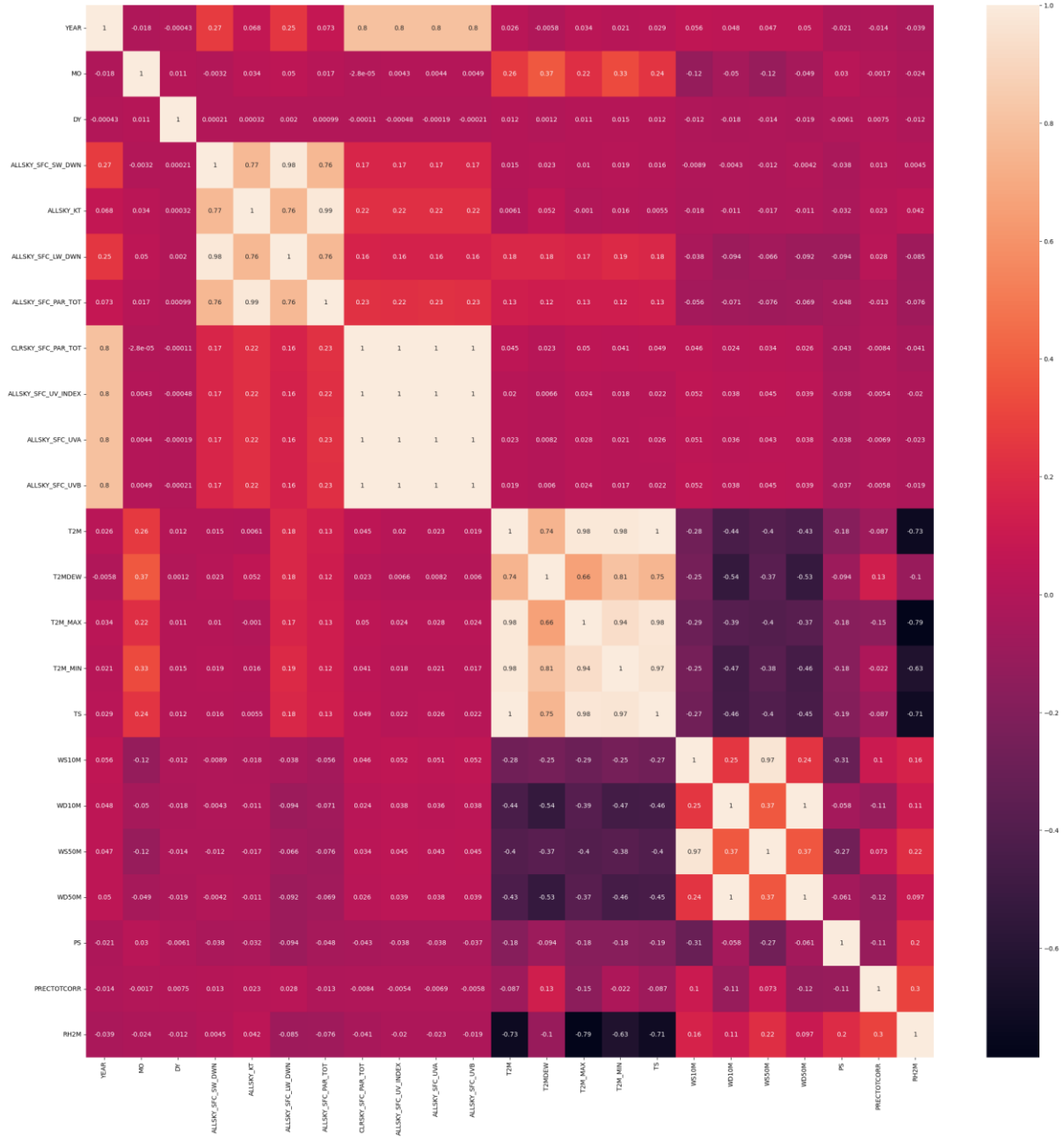
- Clear Sky Surface PAR Total (W/m^2)
- All Sky Surface UV Index (dimensionless)

- All Sky Surface UVA Irradiance (W/m^2)
- All Sky Surface UVB Irradiance (W/m^2)
- Temperature at 2 Meters (C)
- Dew/Frost Point at 2 Meters (C)
- Temperature at 2 Meters Maximum (C)
- Temperature at 2 Meters Minimum (C)
- Earth Skin Temperature (C)
- Wind Speed at 10 Meters (m/s)
- Wind Direction at 10 Meters (Degrees)
- Wind Speed at 50 Meters (m/s)
- Wind Direction at 50 Meters (Degrees)
- Surface Pressure (kPa)
- Precipitation Corrected (mm/day)
- Relative Humidity at 2 Meters (%)

2.3 Data Visualization

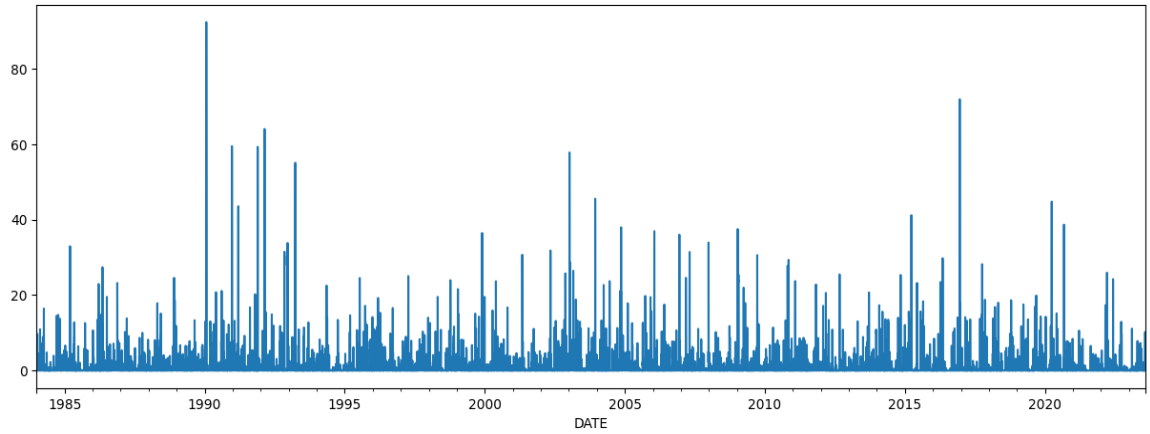
Data visualization is a key step in any project as it helps us to identify potential factors affecting our model and predictions. Following are some key visualizations to better understand the data.

Some key visualizations include histograms and box plots to understand the distribution and outliers of numerical variables, scatter plots to reveal relationships between variables, and bar charts to visualize categorical data. Heatmaps, time series plots, and correlation matrices aid in uncovering complex patterns, while geospatial maps offer spatial insights.

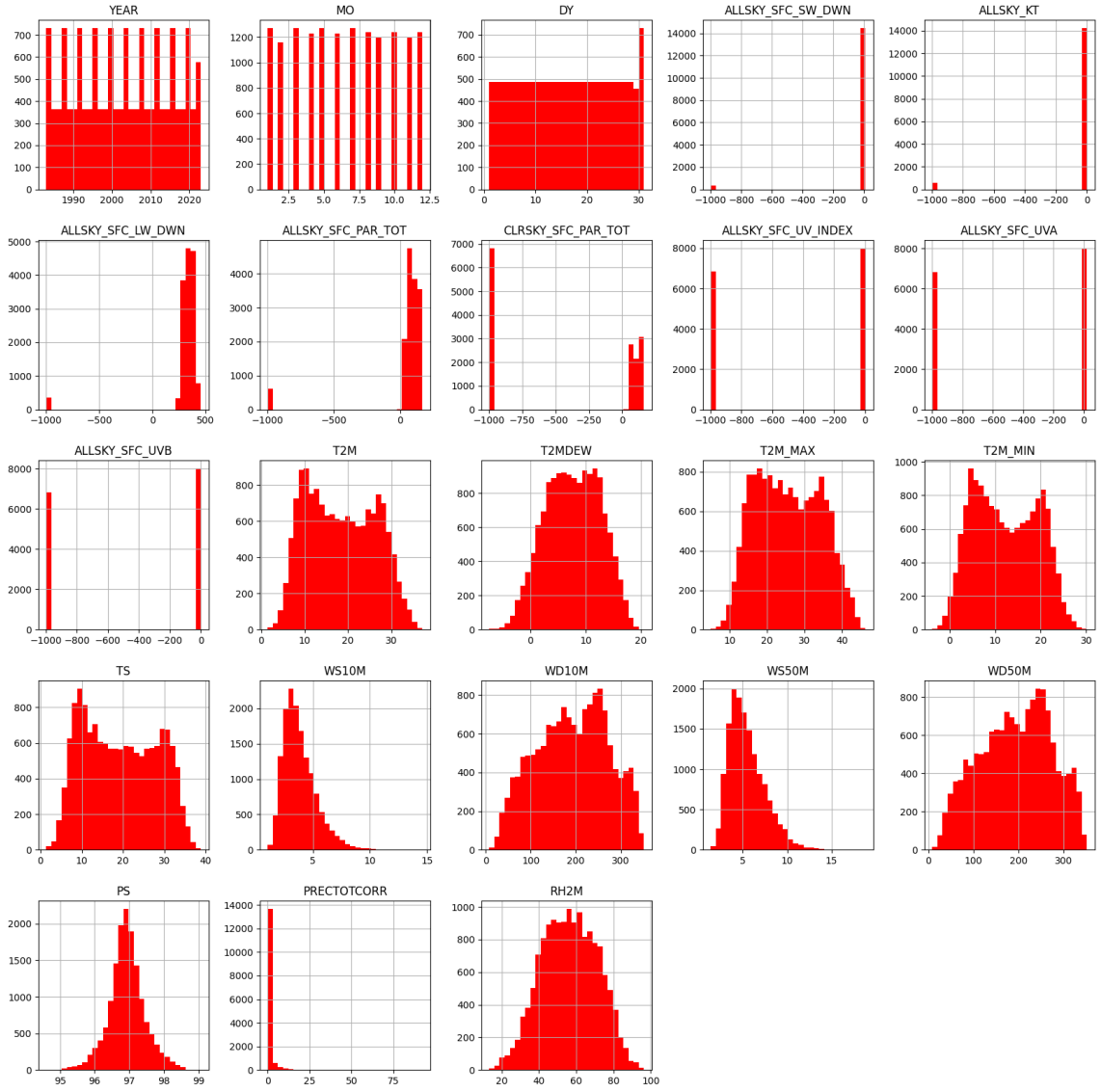


The provided correlation matrix offers a comprehensive overview of the relationships among our features. This matrix plays a crucial role in identifying the features that exhibit direct and significant associations with our target variable, thereby aiding us in understanding the predictive factors

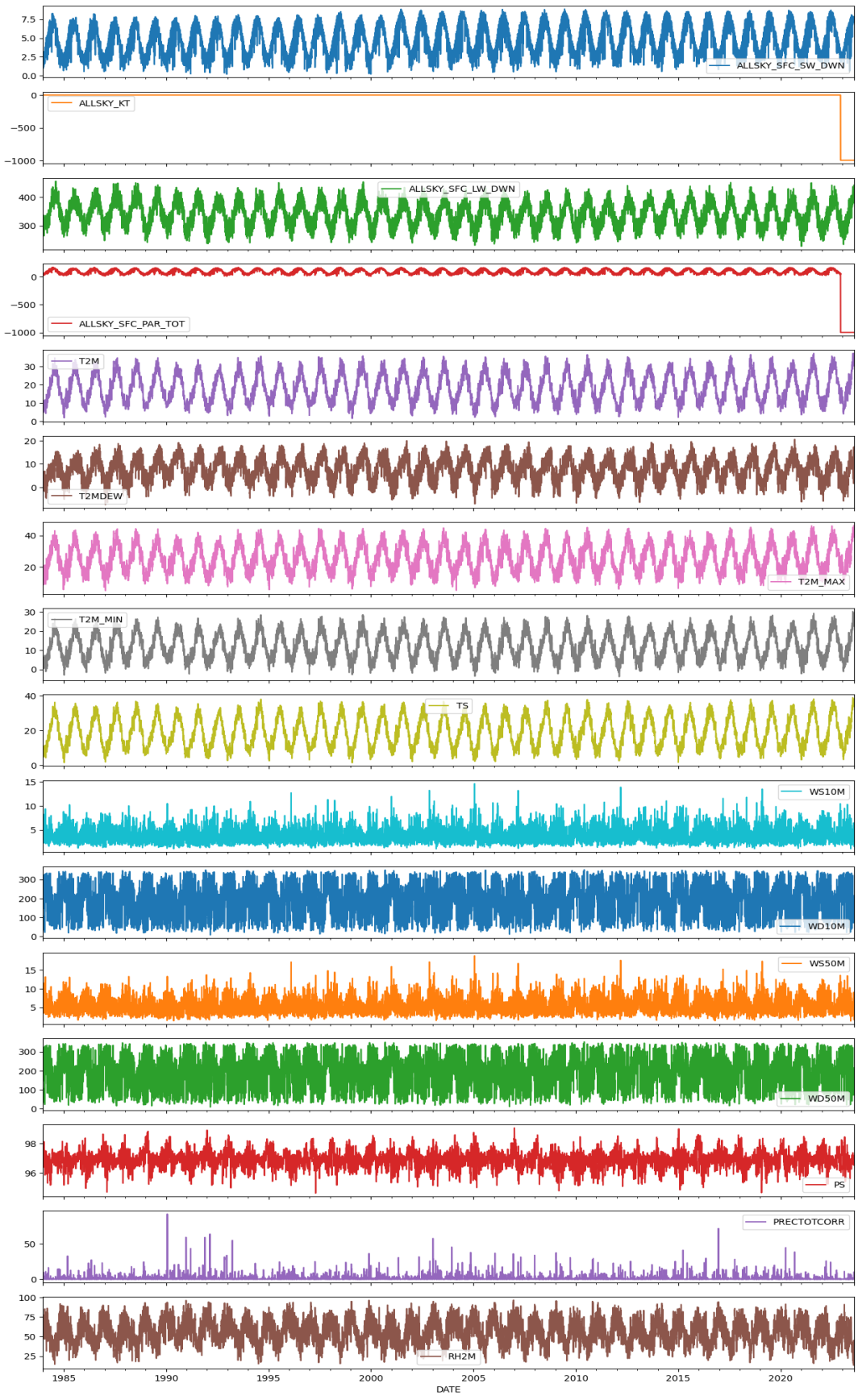
Below graph depicts the variation of precipitation over the years.



Below are 28 histograms indicating all possible features. This helps us find features with extreme values that can be dropped.



All features time series curve



2.4 Statistical Analysis (Checking Stationarity of Series)

Augmented Dickey - Fuller Test

The ADF test belongs to a category of tests called 'Unit Root Test', which is the proper method for testing the stationarity of a time series.

The presence of a unit root means the time series is non-stationary. Besides, the number of unit roots contained in the series corresponds to the number of differencing operations required to make the series stationary. [5]

Null Hypothesis (H0): alpha=1

$$y_t = c + \beta t + \alpha y_{t-1} + \phi \Delta Y_{t-1} + e_t$$

here,

- y(t-1) = lag 1 of time series
- delta Y(t-1) = first difference of the series at time (t-1)

When the test statistic is lower than the critical value shown, you reject the null hypothesis and infer that the time series is stationary.

```
Results of Dickey-Fuller Test:
Test Statistic                  -47.440625
p-value                           0.000000
#Lags Used                        4.000000
Number of Observations Used    14453.000000
Critical Value (1%)              -3.430803
Critical Value (5%)              -2.861740
Critical Value (10%)             -2.566876
dtype: float64
None
```

From above results we can see that the test statistic is much lower than the 1% critical value which implies that the series is stationary for a lag of 30 days.

We notice that the p-value here is close to zero which would indicate that our inference is wrong however keeping in mind the size of the data (14453 samples) we can expect very low p - values.

Note: - Ideally we should check for lag of 365 days in our case if we want to check seasonality year wise however with longer lag values , the model tends to depict increased complexity which consumes a lot of computational power. Since I had a limited computational RAM for my project, I was just able to observe the ADF test for a lag value of 30, for higher values my python notebook crashed.

Kwiatkowski-Phillips-Schmidt-Shin Test (KPSS)

KPSS test is a statistical test to check for stationarity of a series around a deterministic trend. Like ADF test, the KPSS test is also commonly used to analyse the stationarity of a series. However, it has couple of key differences

compared to the ADF test in function and in practical usage.

The KPSS test, short for, Kwiatkowski-Phillips-Schmidt-Shin (KPSS), is a type of Unit root test that tests for the stationarity of a given series around a deterministic trend.

A key difference from ADF test is the null hypothesis of the KPSS test is that the series is stationary.

So practically, the interpretation of p-value is just the opposite to each other.

That is, if p-value is < significance level (say 0.05), then the series is non-stationary. Whereas in ADF test, it would mean the tested series is stationary.

```
Results of KPSS Test:
Test Statistic            0.341577
p-value                   0.100000
Lags Used                29.000000
Critical Value (10%)      0.347000
Critical Value (5%)       0.463000
Critical Value (2.5%)     0.574000
Critical Value (1%)       0.739000
dtype: float64
```

From our previous test results we can observe that p-value (0.10>>>0.05) which implies that the series is stationary. Note that the lag value here is 29 because it was auto selected by the algorithm. By using "nlags=auto," you let the test decide the optimal number of lags, which is often a suitable choice for practical applications. It helps ensure that the test performs well without the need for manual tuning of the lag parameter.

3. METHODOLOGY

3.1 Neural Prophet (Time Series Forecasting)

Neural Prophet is a forecasting tool designed to predict time series data. It is an extension of the popular open-source forecasting library, Prophet, developed by Facebook. Neural Prophet, as the name suggests, incorporates neural networks into the forecasting process, making it more flexible and potentially more accurate for certain types of time series data compared to traditional statistical methods like ARIMA or Prophet.

The original Prophet model, developed by Facebook's Core Data Science team, is known for its simplicity and ease of use, making it accessible to non-experts in time series forecasting. It's capable of handling daily observations with seasonal patterns, holidays, and events. Neural Prophet builds upon this foundation by using a neural network architecture to capture more complex patterns and dependencies in the data.

a) Cross Validation

Time-series cross-validation is a technique that is also referred to as a rolling origin backtest. It involves dividing the data into several folds. * During the first fold, we train the model on a portion of the data and then evaluate its performance on the next set of data points, which are

determined by the fold_pct parameter (percentage of

samples in each fold). * In the next fold, we include the evaluation data from the previous fold in the training data and then evaluate the model's performance on a later set of data points. * This process is repeated until the final fold, where the evaluation data reaches the end of the available data. Essentially, the forecast origin "rolls" forward as we move from one-fold to the next.

b) Visualization

From our dataset we can observe a seasonal trend of relative humidity which also has the highest correlation with the precipitation feature.

RH2M	0.298684
T2MDEW	0.134049
WS10M	0.101536
WS50M	0.072807
ALLSKY_SFC_LW_DWN	0.065793
ALLSKY_KT	0.017259
T2M_MIN	-0.023191
ALLSKY_SFC_PAR_TOT	-0.037191
TS	-0.088917
T2M	-0.089322
PS	-0.109798
WD10M	-0.115651
WD50M	-0.118293
T2M_MAX	-0.156441
ALLSKY_SFC_SW_DWN	-0.219916

For our dataset we used cross validation for 1000 epochs over 3 folds. We can observe the following seasonality and trend from below charts -

c) Result

Following are the test metric scores of all 3 folds on which 1000 epochs were trained -

Test metric	DataLoader 0
Loss_test	0.08958884328603745
MAE_val	1.0714070796966553
RMSE_val	2.8750457763671875
RegLoss_test	0.0

Test metric	DataLoader 0
Loss_test	0.10256016254425049
MAE_val	1.1802048683166504
RMSE_val	3.1289405822753906
RegLoss_test	0.0

Test metric	DataLoader 0
Loss_test	0.07206710427999496
MAE_val	0.9598381519317627
RMSE_val	2.535226583480835
RegLoss_test	0.0

3.2 Random Forest with GRIDCV

A Random Forest Regressor is a machine learning algorithm that leverages ensemble learning to perform regression tasks. It is an extension of the Random Forest algorithm, utilizing multiple decision trees to make predictions. The "random" in Random Forest arises from both feature selection, where a random subset of features is considered at each tree split to mitigate overfitting, and bootstrapping, where random subsets of the training data are sampled with replacement, creating diverse training datasets for each tree.

The algorithm's strength lies in its ability to handle complex and noisy datasets, while reducing overfitting. It aggregates predictions from individual trees to make a final prediction, typically through averaging for regression tasks, offering a robust and versatile tool in machine learning.

```
RandomForestRegressor
RandomForestRegressor(max_depth=7, max_features='sqrt', n_estimators=500,
                      random_state=18)
```

Moving on to the random forest model I used 5-fold CV and tried out various parameters with GRID and obtained the following parameters for the best model -

```
MSE is 0.9545788706067814
RMSE is 0.9770255219833213
```

3.3 XGBoost with GRIDCV

The XGBoost Regressor, an extension of the XGBoost algorithm, is a highly effective machine learning tool for regression tasks. It excels in predictive accuracy by utilizing an ensemble of decision trees and an optimized gradient boosting framework. XGBoost employs a unique combination of regularization techniques, handling missing data, and parallel computation, making it one of the top choices for structured data and tabular data regression problems. It is lauded for its exceptional speed, scalability, and robustness, and it often outperforms other regression algorithms, making it a valuable asset in the toolkit of data scientists and machine learning practitioners.

For our testing purpose I selected the following grid and the results are shown below -

```
grid = {
    'n_estimators': [200, 300, 400, 500],
    'learning_rate': [0.01, 0.1, 0.2, 0.3],
    'max_depth': [3, 4, 5, 6, 7],
    'random_state': [18]
}
```

```
best_xgb_model = CV_xgb.best_estimator_
best_xgb_model
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.2, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=6, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=400, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=18, ...)
```

```
MSE is 0.0040709915013844 39
RMSE is 0.06380432196477319
```

3.4 Multivariate Time Series LSTM

Multivariate Long Short-Term Memory (LSTM) models for rainfall prediction are a powerful approach to forecast rainfall based on multiple input variables or features. Multivariate LSTM models leverage the capability of LSTMs to capture temporal dependencies and patterns in sequential data while considering multiple input variables, such as temperature, humidity, wind speed, and more.First I merged DATE , YEAR and MONTH column into one column feature forming yyyy/mm/dd format.

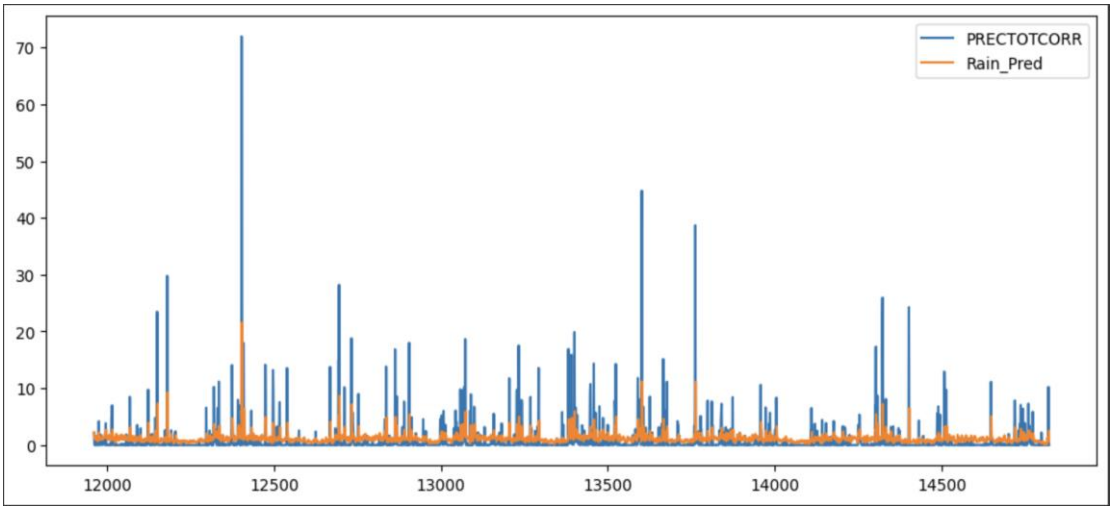
The final LSTM architecture looked something as follows -

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 30, 128)           68096

 leaky_re_lu (LeakyReLU)     (None, 30, 128)           0

 lstm_1 (LSTM)               (None, 30, 128)           131584

 leaky_re_lu_1 (LeakyReLU)   (None, 30, 128)           0

 dropout (Dropout)           (None, 30, 128)           0

 lstm_2 (LSTM)               (None, 64)                49408

 dropout_1 (Dropout)         (None, 64)                0

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 249153 (973.25 KB)
Trainable params: 249153 (973.25 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In the context of LSTM (Long Short-Term Memory) models, "window length" typically refers to the number of time steps or observations that the model considers at once as input. The window length, also known as the sequence length, plays a crucial role in how the model processes the input data.

```
  model.evaluate_generator(test_generator, verbose = 0)
[0.0009866614127531648, 0.013758311979472637]
```

In our case window length was taken as 30 as a month contains on an average 30 days so I found out to have the best results for this value. Below graph shows our predicted rainfall values v/s actual values and we can see that our model performs better as compared to previous models.

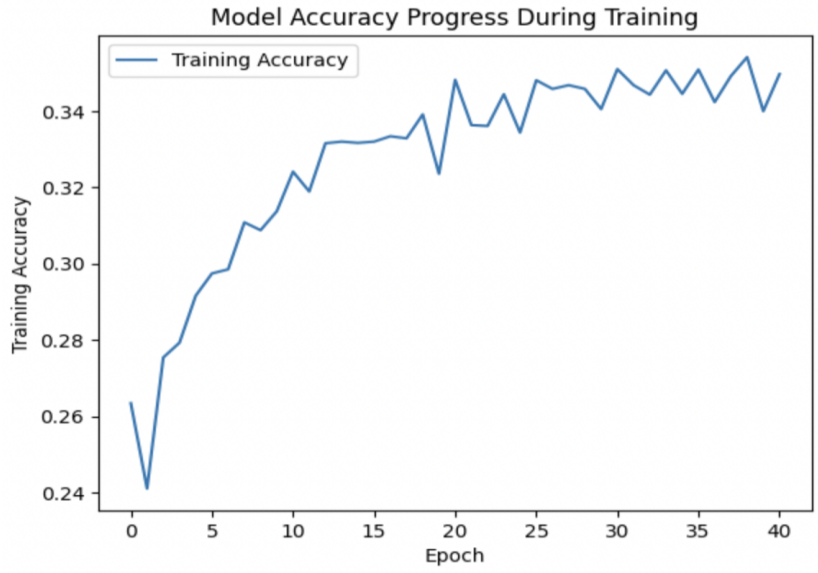


3.5 Predicting with Deep Learning Model

Built, trained and tested a deep learning model. Used cross validation and early stopping methods to avoid overfitting however the model was not very efficient with a low accuracy and a highly fluctuating RMSE (root mean square error).

From above graph we can observe that the dense layers are unable to capture temporal patterns and seasonality that is

which is also one of the reasons why LSTM (a type of RNN) provides superior results as compared to basic dense layers.



4. FUTURE WORK

The project can benefit the addition of many additional features to provide better performance and usability.

Following suggestions may be implemented -

1. **Feature Engineering**- Adding and creating important features can help increase model performance. Our model has an imbalanced dataset with around 33% data with zero values, this usually results in decreased model performance. To overcome this problem new features can be engineered. For example - a new feature containing the count of zeroes in a window length can be created which can be then used in out LSTM model.

2. **Create and Deploy Web app on GCP/AWS/Heroku -** After creating a webapp using Flask or Streamlit one can try to deploy the model and webapp on either GCP, AWS or Heroku to make the model finally public for the public community. This would enhance usability and complete an end-to-end machine learning project.

3. **Predict different factors -** With the model already predicting precipitation we can also try to predict the growth of olive crops. This would however require additional data which includes the branching type of trees, fertilizers used, depth of seed sown for the olive crop, gap between crops etc. Help from industry and farmers is required for this implementation.

5. REFERENCES

[1] Forecasting Global Developments and Challenges in Olive Oil Supply and Demand: A Delphi Survey from Spain

[2] Impact of Climate Change on Olive Crop Production in Italy

[3] Flower and pollen production in the 'Cornicabra' olive (Olea europaea L.) cultivar and the influence of environmental factors

[4] https://morocco-gold.com/growing-morocco-gold-extra-virgin-olive-oil-olives/

[5] https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/?utm_content=cmp-true