Weather Prediction Using Machine Learning

PROBLEM STATEMENT

"Weather Prediction Using Machine Learning" implements a supervised, regression machine learning problem of Weather Prediction using a real time dataset from https://www.ncdc.noaa.gov/cdo-web/. It aims to forecast the trend of maximum temperature of a day based on 6 years of historical weather data which is used to train and test the models.

The input to the program is a real time dataset as specified above. The dataset will be used to train the models using various machine learning regression algorithms like Random Forest, Support Vector Machine, Gradient Boosting and Logistic Regression. After producing prediction results the project aims to study the impact of reducing the number of features to only the important features and again producing prediction results with reduced number of features.

For some models, decreasing the number of features to only the important ones can increase performance and therefore should be done. However, in other situations, performance will decrease but run time will also decrease. The final decision on how many features to retain will therefore be a trade-off between accuracy and run time.

The project implementation will analyze the trade-off between accuracy and run time both analytically and visually. Object Oriented Programming approach is followed that aids the implementation efficiently.

PROBLEM ANALYSIS

The problem specified is a supervised, regression machine learning problem. It is supervised because both the features (historical weather data for the city) and the target (actual maximum temperature for the day) that is to be predicted are available. During the training phase, the machine learning models based on various machine learning algorithms are given both the features and targets and it must learn how to map the data to a prediction. Moreover, this is a regression task because the target value is continuous, as opposed to discrete classes in classification.

SPECIFICATIONS

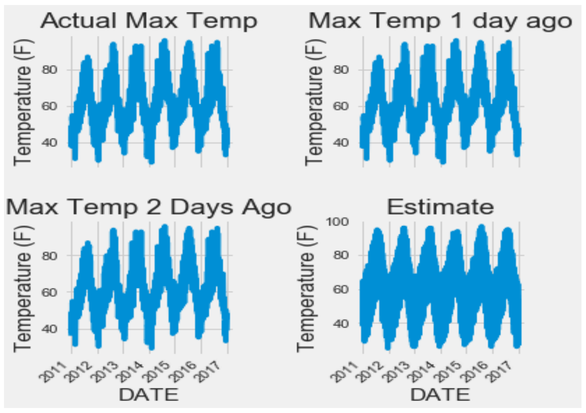
To ensure a realistic example the dataset is retrieved from the NOAA climate data online tool, the reference to which is provided in section 1. The input dataset specifies 12 variables in which the column “actual” represents the target value. The remaining 11 variables are the features whose mapping with the target value is analyzed by the machine learning models during training in order to make efficient predictions of the target when the models are tested on unseen data. The dataset variables are as below:

- **year:** year ranging from 2011 to 2017; **month:** number of month of the year
- **day:** number of day of the month; **weekday:** day of the week as a character string
- **ws_1:** wind speed 1 day ago the target day; **prcp_1:** precipitation level 1 day ago the target day
- **snwd_1:** snow depth 1 day ago the target day; **temp_ 1:** max temp 1 day ago the target day
- **temp_2:** max temp 2 days ago the target day; **estimate:** estimate of max temp of the target day
- **average:** Historical average max temp of the target day
- **actual:** The actual max temperature of the target day

FLOW OF THE PROGRAM

The program starts by loading the dataset mentioned earlier as the pandas Data Frame. The data frame is used to understand the dataset shape, distribution of the features, the missing values etc. As in any machine learning problem it is important to study trends of the features, co-relations between them and their relations with the target value, the visual graphs are plotted using matplotlib and seaborn to get a clear idea of the trends, co-relations of features and relationship with the target value as depicted in Fig [1], Fig[2] and Fig[3].

Followed by the data visualization, the program prepares the data for machine learning models. Different machine learning models like Random Forest, Gradient Boosting, Logistic Regression and Support Vector Machine are fed by the training data (the features and the target value) in which they learn the mapping of features to the target value. During testing phase, the models make predictions for the actual maximum temperature for the day based on unseen data.



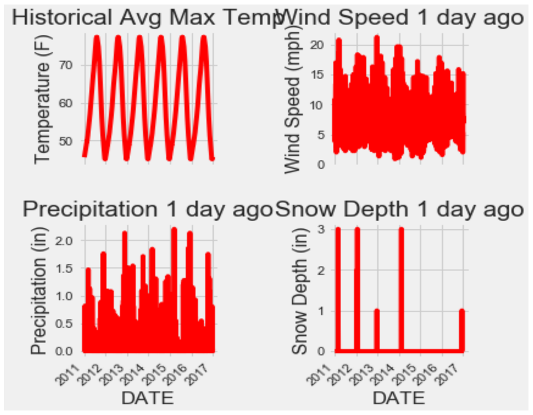


Figure 1. Feature Trends Figure 2. Feature Trends

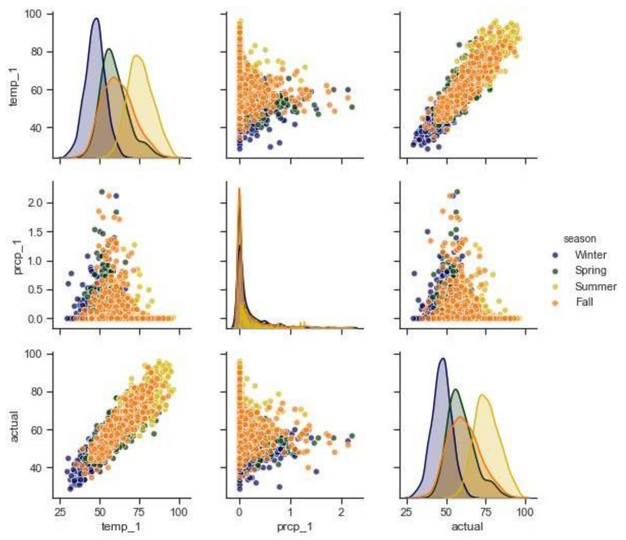


Figure 3. Features & Target Co-Relations using Pair-Plot

The program defines a baseline degree of error that must be exceeded by the prediction model. The baseline error is defined as average of (Historical average max temp of the target day - The actual max temperature of the target day). The baseline error corresponds to the situation when the prediction model every time predicted the actual maximum temperature of the day to be its Historical average max temp only. The model with the best comparative results in terms of error degree is chosen for further analysis by the program, say *"reference_model"* for ease in the report.

The important features are computed analytically and visually for the *"reference_model"*. The important features are the ones on which the *"reference_model"* relied more to predict the actual maximum temperature of the day among all features that were provided to it as an input.

After reducing the number of total features to only the important features, the *"reference model"* produces prediction results again but this time the input is only the important features.

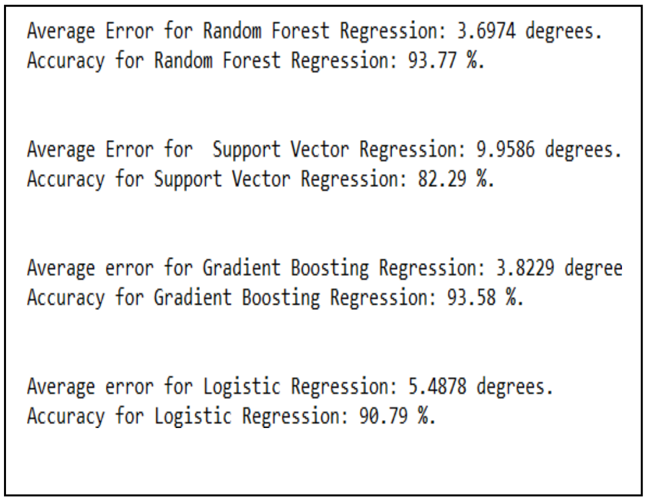
For some models, decreasing the number of features can increase performance and therefore should be done. However, in other situations, performance will decrease but run time will also decrease. The final decision on how many features to retain will therefore be a trade-off between accuracy and run time.

Therefore, the total average run time between training and testing phase of the model is computed analytically and visually with total features and the reduced features (Important Features) to analyze how much is the impact of reducing features on the computation time.

Different class structures are implemented in the files that constitute the methods to achieve the flow of the program.

TESTING

To test the software the script *main_weather_prediction.py* needs to be run. The program starts with data visualization as depicted in **figure 1**, **figure 2** and **figure 3**. The average baseline error is computed as **4.79** followed by the machine learning model predictions as in **figure 4.** The prediction results remain the same every time the script is run because the state is saved in random_state=42 which initializes the internal random number generator for train-test sets.



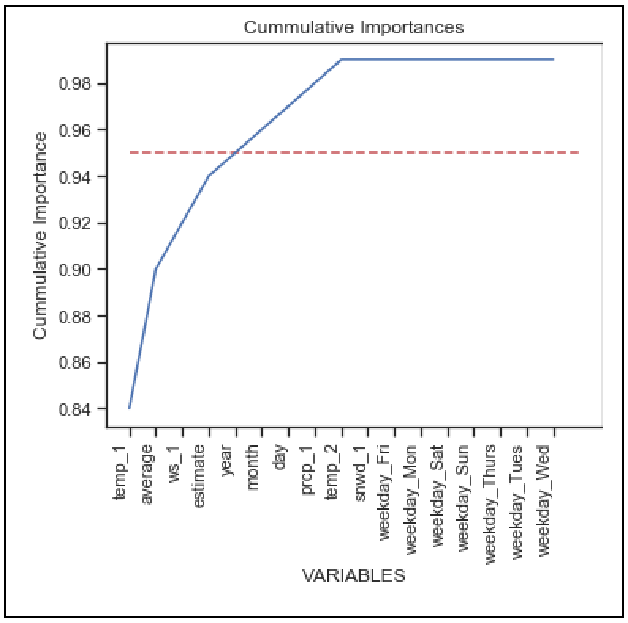
```
Average Error for Random Forest Regression: 3.6974 degrees.
Accuracy for Random Forest Regression: 93.77 %.


Average Error for  Support Vector Regression: 9.9586 degrees.
Accuracy for Support Vector Regression: 82.29 %.


Average error for Gradient Boosting Regression: 3.8229 degree
Accuracy for Gradient Boosting Regression: 93.58 %.


Average error for Logistic Regression: 5.4878 degrees.
Accuracy for Logistic Regression: 90.79 %.
```

Figure 4. Machine Learning Model Predictions

Figure 4. depicts the better results produced by Random Forest Regression model, therefore it becomes the *"reference model"*.

After choosing the reference model, program computes the important features on which the Random Forest Model relied comparatively more to make predictions.

The cumulative importance of features is visualized as in **Figure 5** and only those features are retained that impacted the model's prediction up to **95%**.



It can be inferred from Figure 5 that only the initial 5 variables on the x-axis contributed to 95% of decision making, hence only those features are taken as the important features.

The prediction results are computed again on the *reference model* with reduced number of features. It is certainly expected that reducing to important features give better accuracy results but taking the initial 5 features deteriorates the prediction accuracy to **93.64%** as compared to **93.77%** for total features.

Figure 5. Cumulative Importance of Features

If the accuracy results deteriorate by reducing the number of features, the approach can be to change the number of features retained, either increase or decrease as per the feature relevance by understanding but should be less than the total features. If the results improve it should be done however if it deteriorates again like in this program (when taking initial 5 features accounting for 95% of the importance) a tradeoff must be made between accuracy and run-time.

The program takes an average of the run time between training and testing when fed with the total features and with the reduced features that accounted for 95% of the importance. The accuracy decreases but the run-time also decreases significantly by reducing the number of features.

For the models, decreasing the number of features can increase performance and therefore should be done. However, in other situations, performance will decrease but run time will also decrease significantly like the one presented here. The final decision on how many features to retain will therefore be a trade-off between accuracy and run time. In this case the comparison between accuracy results and run time for retaining all features and only the ones accounting for 95% of importance are depicted in **Figure 6.**

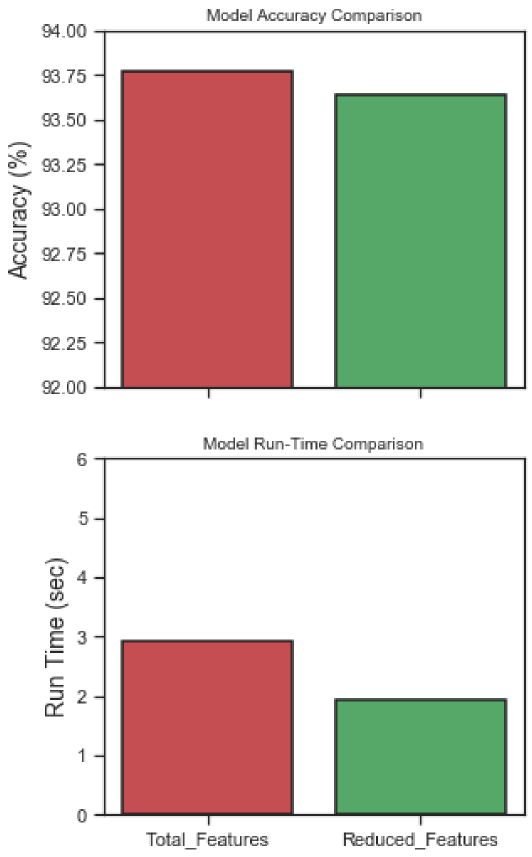


Figure 6. Comparing results for Total Features and Reduced Features

IMPACT

The project implementation gives analytical and visual understanding of the weather prediction and explains why it is necessary to reduce the model to only the important features. Reducing the features can improve performance and decrease run time however if the performance decreases the number of important features can be retained based on the trade-off between accuracy and run-time.

The object-oriented approach allows a clean design of the program where all functionalities can be accessed and even changed without changing the overall design of the program. This enables a flexibility to use any method to solve a part of the program.