Carlos Ayala Medina A01703682

Rain prediction with decision tree and random forest

Abstract

This document presents the implementation of a decision tree and random forests algorithms for binary classification. To predict if there will be rain based on the current weather conditions. The dataset used in this project includes historical weather data from multiple cities in Australia with features such as temperature, wind speed, cloudiness, etc.

Introduction

Weather prediction is a crucial aspect of meteorology that has significant implications for various industries such as agriculture and transportation. Traditional weather prediction methods rely on physical models and numerical simulations that require extensive computational resources and can be time consuming. Now we can use machine learning to predict the weather in a more efficient way.
Here we'll see the implementation of two techniques, decision trees and random forest to predict if there will be rain the next day.

State of the art

In recent years machine learning has improved current technology of weather prediction. In 2021 Jonathan A. Weyn (1) developed a convolutional neural network to improve his seasonal forecasting. His system is called Deep Learning Weather Prediction (DLWP) and is trained with past weather data which makes it differ from regular weather prediction models that create mathematical representations of physical laws.

Although his model does not perform better that current existing models it is computationally more efficient that other approaches since it only require 3 seconds to compute. Which shows promise for machine learning techniques to be commonly used in weather prediction. So, we'll be using two techniques in this project.

Decision trees and random forests are popular machine learning techniques that can be applied to a multitude of scenarios. Decision trees are an algorithm that can be considered simple and interpretable that works for classification and regression problems. It's like its name says a tree-

like model that recursively checks the input data based on a set of conditions and rules until a stopping criterion is met. These rules are based on the features from the input dataset.

A decision tree has to components, nodes and leaf's. A node represents the features and conditions of the input data. The leaves represent the class labels or regression values that you'll receive at the end of the tree.

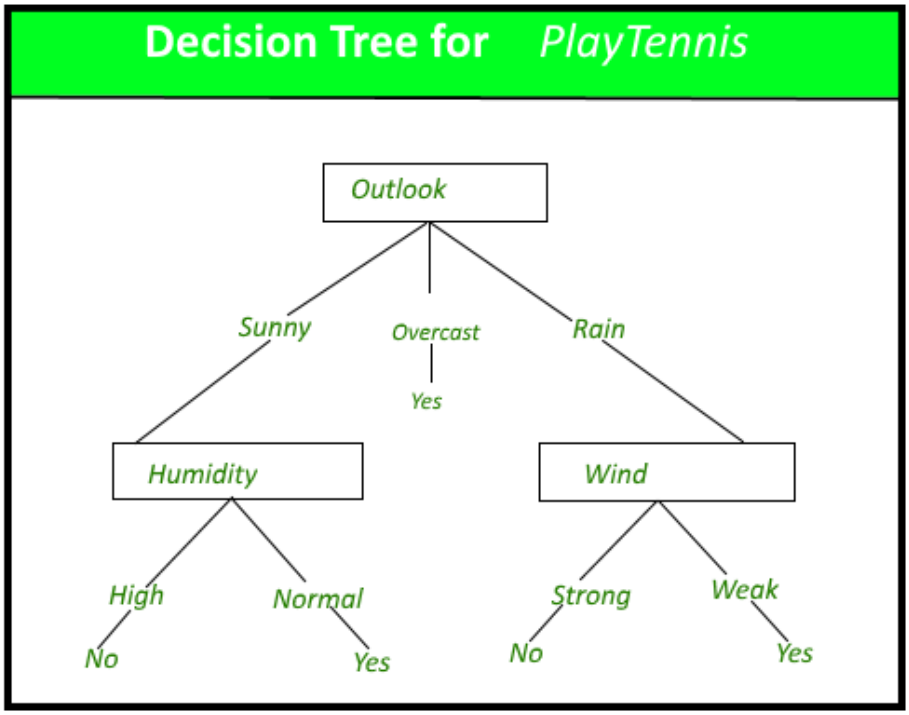


Figure 1: Decision tree from GeeksforGeeks

The tree is constructed by selecting the features that best separate the input data and recursively splitting the data based on these features. The splitting will continue until it reaches a stopping criterion like the maximum depth allowed.

Random forests are an ensemble learning technique that is also used for classification and regression. It can be considered as an extension of decision trees since is based on the idea of constructing multiple decision trees and combining their outputs to make a final prediction.

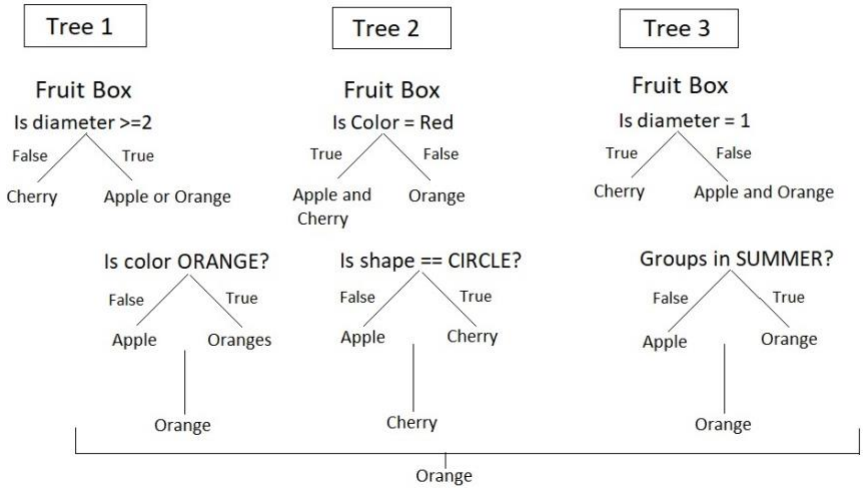


Figure 2: Random Forest from GeeksforGeeks

Random forest has several advantages over individual decision trees, like improved accuracy, robustness to noise, and they are less prone to overfitting, especially when the number of threes in the forest is large.

They counter overfitting since they introduce randomness into the modeling process. A random forest will randomly select different subsets of the features to train each tree, with all trees relying in different features the likelihood of overfitting is reduced. Also, the final prediction which is a combination of all individual predictions helps smooth out any noise and errors.

Dataset

The dataset we'll be using in this project contains about 10 years of daily weather observations from numerous Australian weather stations. It is a recompilation of multiple datasets provided from the Bureau of Meteorology of the Australian Government found of Kaggle.

It has 145,461 rows and 23 columns such as location, temperature, rainfall, cloudiness, wind speed, etc. Our target for this project will be "RainTomorrow" that based on the current weather will predict if there is going to rain the next day.

Data Cleaning

Columns:

- Date
- Location
- MinTemp
- MaxTemp
- Rainfall
- Evaporation
- Sunshine
- WindGustDir

- WindGustSpeed
- WindDir9am
- WindDir3pm
- WindSpeed9am
- WindSpeed3pm
- Humidity9am
- Humidity3pm
- Pressure9am
- Pressure3pm
- Cloud9am
- Cloud3pm
- Temp9am
- Temp3pm
- RainToday
- RainTomorrow

The marked features in gray are dropped because some are not relevant like date and location. The rest are dropped only because they are categorical features and while they can be encoded using **OneHotEncoder**, it is computationally expensive, and it will not make any sense since these features are not ordinal.

The red features values are replaced by 1 and 0.

```
# Data cleaning
df.columns = ["Date", "Location", "MinTemp", "MaxTemp", "Rainfall", "Evaporation", "Sunshine", "WindGustDir", "WindGustSpeed",
              " WindDir9am", "WindDir3pm","WindSpeed9am", "WindSpeed3pm", "Humidity9am", "Humidity3pm", "Pressure9am",
              "Pressure3pm", "Cloud9am", "Cloud3pm", "Temp9am", "Temp3pm", "RainToday", "RainTomorrow"]
df['RainToday'] = df['RainToday'].replace('No', 0)
df['RainToday'] = df['RainToday'].replace('Yes', 1)
df['RainTomorrow'] = df['RainTomorrow'].replace('No', 0)
df['RainTomorrow'] = df['RainTomorrow'].replace('Yes', 1)
df = df.dropna()
df_x = df[["MinTemp", "MaxTemp", "Rainfall", "WindGustSpeed", "WindSpeed9am", "WindSpeed3pm",
           "Humidity9am", "Humidity3pm", "Pressure9am", "Pressure3pm", "Temp9am", "Temp3pm", "RainToday"]]
df_y = df[["RainTomorrow"]]
```

Figure 3: Data Cleaning code

Implementation

We used Python's scikit-learn library for training and evaluating the models.

```
DecisionTreeClassifier
RandomForestClassifier
```

Feature selection

The features that weren't dropped on the data cleaning phase will be used to train our model. With the next correlation matrix, we can see the relation between variables.

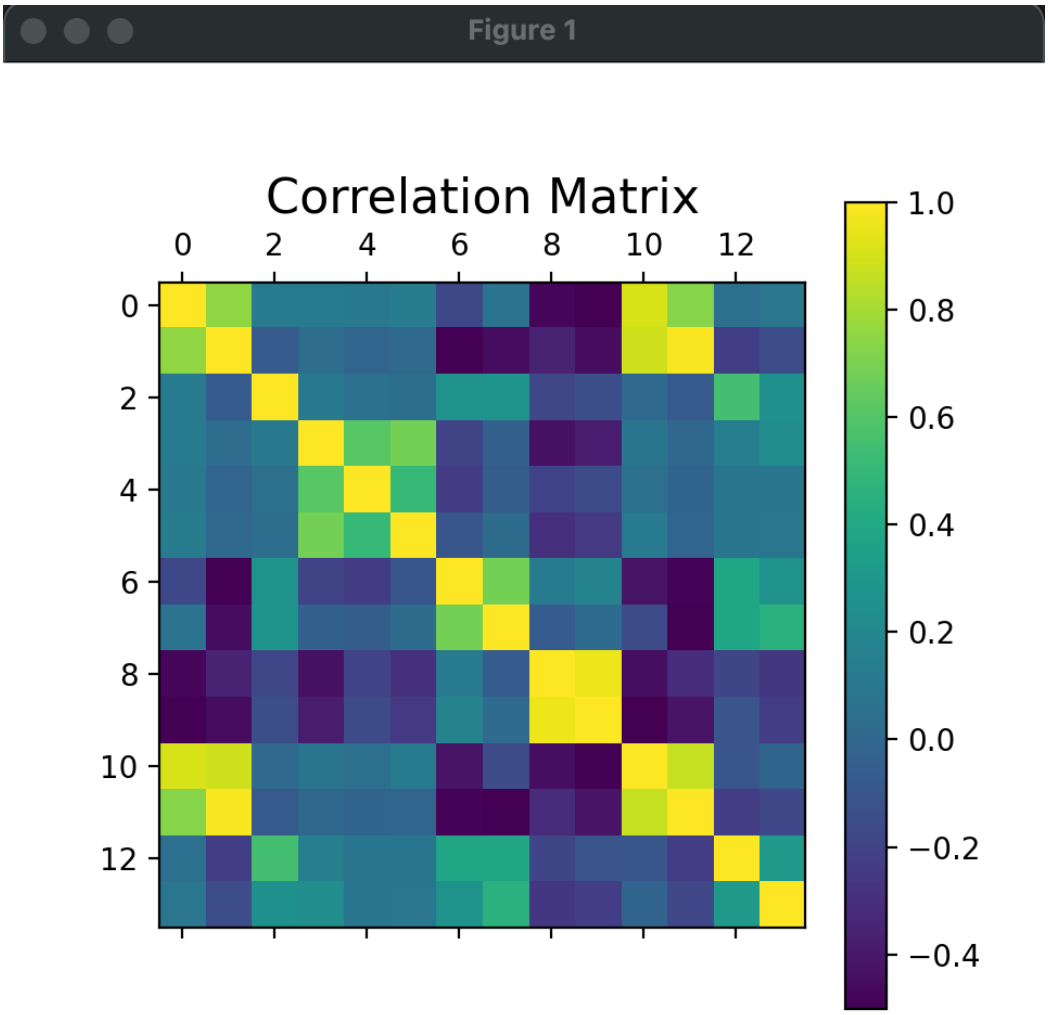


Figure 4: Correlation Matrix

There are not many strong relations in this dataset, but most features have a positive (above 0) correlation with our target feature, so all features are used in this implementation.

Data Split

We are using 80% of the dataset for training and 20% for testing.

```
# Split data
X_train, X_test, y_train, y_test = train_test_split(df_x, df_y,
                                                    test_size=0.2,
                                                    random_state=0)
```

Figure 5: Data Split code

Decision Tree

```
clf = tree.DecisionTreeClassifier(random_state=0, max_depth=8)

clf = clf.fit(X_train, y_train)

# Make predictions using the testing set
y_pred = clf.predict(X_test)

# Accuracy
accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy_score(y_test, y_pred))

# Correlation Matrix
plt.matshow(df_corr.corr())
cb = plt.colorbar()
plt.title('Correlation Matrix', fontsize=16)
plt.show()
```

Figure 6: Decision Tree Classifier code

In Figure 6 we can see the Decision Tree code. It includes the accuracy score and the correlation matrix.

Random Forest

```
clf = RandomForestClassifier(max_depth=16, random_state=0)
clf = clf.fit(X_train, y_train.values.ravel())

# Get the trees for visualization
estimator = clf.estimators_[99]

# Make predictions using the testing set
y_pred = clf.predict(X_test)

# Accuracy
print(accuracy_score(y_test, y_pred))
```

Figure 7: Random Forest code

In Figure 7 we can see the Random Forest code. It includes the extraction of a tree (estimator) for visualization.

```
# Visualization, requires Graphviz, install it with pip install graphviz
# Prints the extracted tree to a pdf file
dot = tree.export_graphviz(estimator, out_file=None,
                           feature_names=["MinTemp", "MaxTemp", "Rainfall", "WindGustSpeed", "WindSpeed9am", "WindSpeed3pm",
                                          "Humidity9am", "Humidity3pm", "Pressure9am", "Pressure3pm", "Temp9am", "Temp3pm",
                                          "RainToday"],
                           class_names="RainTomorrow",
                           filled=True, rounded=True,
                           special_characters=True,
                           max_depth=3)
graph = graphviz.Source(dot)
graph.view()
```

Figure 8: Visualization code

In Figure 8 we can see the code for the visualization of the tree. In the forest case we use the estimator we extracted previously. It is required to install graphviz.

```
pred = clf.predict(new_df)
if (pred[0] == 1):
    print('It is going to rain tomorrow')
    print("`、ヽ`ヽ`、ヽ(ノ><)ノ `、ヽ`☂ヽ`、ヽ")
else:
    print('No Rain Tomorrow')
    print("\\\ (ーー゛) //*")
exit = input('exit? (write no to continue): ')
```

Figure 9: Prediction code

In Figure 9 we can see the code for the prediction of rain.

Conclusion and results

In the case of the Decision Tree (with a max depth of 8), we got an accuracy of:

0.8394186458702588

```
(env) (base) Rain-prediction-Trees-Forest$ python tree.py
Accuracy: 0.8394186458702588
```

With the Random Forest (with a max depth of 16), we got an accuracy of:

0.8597128677773839

```
(env) (base) Rain-prediction-Trees-Forest$ python forest.py
0.8597128677773839
```

There's not much difference in this case between both techniques. I tried changing the depth to see if there was a change in one of the implementations, but it seems that they stay around 0.83 and 0.85 respectively.

I expected a bigger difference in their performance since a forest is supposedly a stronger algorithm. The reason for these similar results can be the weak correlation between the features. Perhaps with a stronger correlation we can see a bigger difference between both.

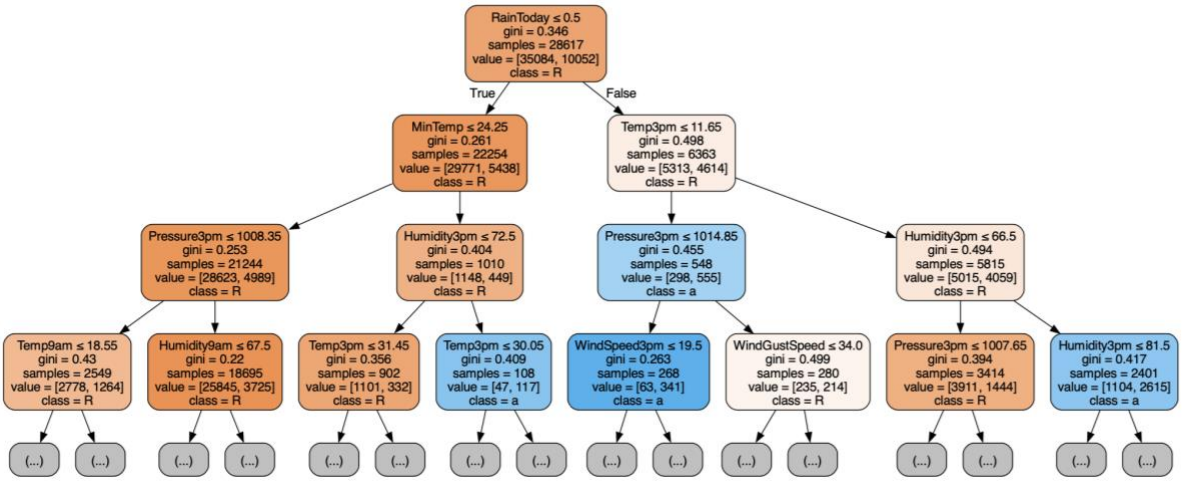
But in conclusion the results are positive, with both techniques we can achieve a good enough accuracy for predicting weather conditions. Machine learning seems to be a plausible technology to be used in forecasting.

Output

The code includes a menu for testing with randomized and manual data.

```
(env) (base) Rain-prediction-Trees-Forest$ python forest.py
0.8597128677773839
Press anything to write your own data or write ran for randomized data: ran
minTemp, maxTemp, rainfall, windGustSpeed, windSpeed9am, windSpeed3pm, humidity9am, humidity3pm, pressure9am, pressure3pm, temp9am, temp3pm, rainToday
[[0.21, 23.69, 13.19, 67.95, 53.39, 44.12, 85.27, 86.29, 990.44, 1015.66, 8.43, 37.86, 0]]
It is going to rain tomorrow
```

It will also return the visualization of the tree (or a tree from the forest) for learning purposes.



References

1. Weyn, J. A., Durran, D. R., Caruana, R., & Cresswell-Clay, N. (2021). Sub-Seasonal Forecasting With a Large Ensemble of Deep-Learning Weather Prediction Models. *Journal of Advances in Modeling Earth Systems*, *13*(7). https://doi.org/10.1029/2021ms002502