CAP5602 Project – Predicting Austrailian Rainfall

Group 5

Venkata Naga Sai Anvitha  
School of Computing and Information Sciences  
Florida International UniversityMiami, Florida  
vtang004@fiu.edu

Anthony Athens  
School of Computing and Information Sciences  
Florida International UniversityMiami, Florida  
aathe004@fiu.edu

Michael Cordero   
School of Computing and Information Sciences  
Florida International UniversityMiami, Florida  
mcord094@fiu.edu

Aseem Sharma  
School of Computing and Information Sciences  
Florida International UniversityMiami, Florida  
ashar097@fiu.edu

*Abstract*—This paper details efforts made in predicting next-day rainfall in Australia when provided 10 years worth of weather observations in the region.

Keywords—Artificial Intelligence, Machine Learning, Data, Preprocessing, Australia, Weather

# Introduction

This paper details Group 5’s efforts in the course project for the fall semester of CAP5602 Introduction to Artificial Intelligence. The problem selected involves attempting to predict whether or not it will rain the next day when provided ten years worth of weather observations across Australia.

This paper will describe the raw data provided, explain the necessary preprocessing techniques executed, and discuss the various experimentation methods performed while attempting to maximize prediction accuracy.

# Related Work

## Pending

Pending

## Pending

Pending.

# Data and Methods

The purpose of this section is to introduce the dataset, detail the preprocessing techniques implemented, and describe the experimentation conducted in attempting to maximize prediction accuracy.

## Dataset Introduction

The purpose of this section is to review the details of the dataset chosen for this project, at a Micro and a Macro perspective. The dataset’s shape and features will be analyzed in this section.

Size & Shape:

We used the Kaggle Dataset whose name is ‘weatherAUS.csv’, which is open source and contains 145,460 records of weather data in Australia from 2008 to 2017 with 23 features in it.

Feature Analysis:

The description of each feature from the dataset is written in the table below which will help to understand the dataset better.

|  |  |  |
| --- | --- | --- |
| **Sl.no** | **Feature Name** | **Feature Description** |
| 1 | Date | The date of observation |
| 2 | Location | The name of the city for which we have information |
| 3 | MinTemp | Minimum temperature recorded of the day |
| 4 | MaxTemp | Maximum temperature recorded of the day |
| 5 | Rainfall | The total amount of rainfall for the day. |
| 6 | Evaporation | It is the Class A pan evaporation (mm) in the 24 hours to 9am |
| 7 | Sunshine | The duration of the day's bright sunshine in hours. |
| 8 | WindGustDir | The direction of the strongest wind gust in the 24 hours to midnight |
| 9 | WindGustSpeed | The speed (km/h) of the strongest wind gust in the 24 hours to midnight |
| 10 | WindDir9am | Direction of the wind recorded at 9am |
| 11 | WindDir3pm | Direction of the wind recorded at 3pm |
| 12 | WindSpeed9am | Wind speed (km/hr) averaged over 10 minutes prior to 9am |
| 13 | WindSpeed3pm | Wind speed (km/hr) averaged over 10 minutes prior to 3pm |
| 14 | Humidity9am | Percentage of Humidity at 9am |
| 15 | Humidity3pm | Percentage of Humidity (percent) at 3pm |
| 16 | Pressure9am | Atmospheric pressure (hpa) reduced to mean sea level at 9am |
| 17 | Pressure3pm | Atmospheric pressure (hpa) reduced to mean sea level at 3pm |
| 18 | Cloud9am | By 9 a.m., a portion of the sky was clouded over. "Oktas," a unit of eigths, are used to measure this. It shows how many sky eigths are blocked by clouds. A score of 0 denotes an entirely clear sky, while an 8 indicates an entirely cloudy sky. |
| 19 | Cloud3pm | At 3 p.m., a portion of the sky was hidden by cloud. |
| 20 | Temp9am | The amount of Temperature (in degrees C) at 9am |
| 21 | Temp3pm | The amount of Temperature (in degrees C) at 3pm |
| 22 | RainToday | It contains a Boolean value 1(Yes) if the amount of precipitation (in mm) in the 24 hours leading up to 9am exceeds 1mm; else, 0 (No) |
| 23 | RainTomorrow | The mm of rain expected the next day. utilized to produce a response variable ‘RainTomorrow’. |

## Preprocessing

Null Values:

The raw data provided in this project required preprocessing before any successful regression could be run. The raw dataset contained 145460 points of Australian weather data but there were several instances of ‘null’ values observed throughout the dataset. The ‘null’ values presented an obvious concern when located in the “RainTomorrow” feature as this is the label we are trying to predict. That said, there were several instances of ‘null’ values observed throughout the dataset in features used to drive prediction success of the of this regression problem. In other words, there were several instances of failed data collection for Temperature, Rainfall, Wind, Sunshine, Humidity, etc. To remedy the missing data condition, a dropna() method was executed on the raw dataframe. The resulting dataframe was reduced to 56,420 rows of cleaned, correctly populated data.

One Hot Encoding:

After the dataset was cleaned, additional steps were required in order to prepare the dataset for regression. Multiple features in the dataset, including “Location”, “WindGustDir”, “WindDir9am”, “WindDir3pm”, and “RainToday”, were categorical in nature. The existing format of these categorical features would have prevented their utilization in any attempted regression models. To remedy this condition, one hot encoding was executed on these features, creating a new binary feature for each categorical value present, indicating the presence of said value in a new feature. The execution of the one hot encoding expanded the number of features from 23 to 92, accounting for all values present in the categorial features mentioned.

Train-Test Split:

After executing the cleaning to address the null values and the required one hot encoding, the format of the dataset was now digested to the point where regression could be run. To Prepare for the regression, execution of a train-test split was required. To accomplish this, we first need to split the dataframe into ‘What’s being predicted?’ vs ‘What’s doing the predicting?’. ‘Y’ was defined as the label being predicted, and was set to the “RainTomorrow” feature from the existing dataframe. ‘X’ was defined as the features doing the predicting, and was set to all other features in the existing dataframe, with the exception of “Date”. After this split was conducted, and additional split was required to enable model training and testing. ‘X’ and ‘Y’ were randomly split into ‘X\_Train’, ‘X\_Test’, and ‘Y\_Train’, ‘Y\_Test’, at an 80:20 Train:Test ratio. Execution of this split enables models to be trained and tested.

## Expiriments and Evaluation

The purpose of this section is to describe the models attempted and discuss the observed accuracy.

Models:

Multiple models were tried while attempting to achieve the highest prediction accuracy. Decision Tree, Logistic Regression(standard), Logistic Regression(liblinear), and SVM were all models attempted that will be reviewed below.

*Decision Tree*

The standard decision tree classifier from the ‘tree’ library was used while building the model in this section and is seen in the code placed below.

model = tree.DecisionTreeClassifier()

model = model.fit(X\_train, Y\_train)

This model yielded 79.9% accuracy on the test set and produced the an AUC of 0.71 and the confusion matrix placed below.

Line chart

Description automatically generated

Treemap chart

Description automatically generated with medium confidence

*Logistic Regression*

The standard Logistic Regression Classifier from the ‘LogisticRegression’ library was used while building the model in this section and is seen in the code placed below. After performing a grid search, setting C=100 was deemed optimal as it produced the highest accuracy.

model = LogisticRegression(C=100)

model = model.fit(X\_train, Y\_train)

This model yielded 85.2% accuracy on the test set and produced the an AUC of 0.88 and the confusion matrix placed below.

Graphical user interface

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with low confidence

*Logistic Regression(liblinear)*

While amending the standard Logistic Regression Classifier from the ‘LogisticRegression’ library, the solver was set to ‘liblinear’ while building the model in this section. After performing a grid search, setting C=1 was deemed optimal as it produced the highest accuracy. The model can be seen below.

model = LogisticRegression(solver = 'liblinear',C=1, random\_state=0)

model = model.fit(X\_train, Y\_train)

This model yielded 85.8% accuracy on the test set and produced the an AUC of 0.89 and the confusion matrix placed below.

Graphical user interface

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with low confidence

*Support Vector Machine – Kernel: RBF*

The Support Vector Machine classifier from the ‘svm’ library was used while building the model in this section. While building this model, the kernel was set to ‘rbf’ and C was set to 1 after performing a grid search to optimize accuracy. The model can be seen below.

model = svm.SVC(kernel='rbf', gamma='auto', C=1.0)

model = model.fit(X\_train, Y\_train)

This model yielded 84.2% accuracy on the test set and produced the an AUC of 0.85 and the confusion matrix placed below.

A picture containing graphical user interface

Description automatically generated

Chart, treemap chart

Description automatically generated

*Support Vector Machine – Kernel: Linear*

The Support Vector Machine classifier from the ‘svm’ library was used while building the model in this section. While building this model, the kernel was set to ‘linear’ and C was set to 1 after performing a grid search to optimize accuracy. The model can be seen below.

model = svm.SVC(kernel='linear', C=1.0)

model = model.fit(X\_train, Y\_train)

This model yielded 85.67% accuracy on the test set and produced the an AUC of 0.89 and the confusion matrix placed below.

A picture containing graphical user interface

Description automatically generated

A screenshot of a computer

Description automatically generated with low confidence

Additional Experiments:

After the assessment of the originally created models, additional experiments were conducted while attempting to improve the accuracy further. The additional experiments conducted were comprised of a mix of entirely new, complex models as well as rerunning the existing models after implementing alternative preprocessing methods. The additional experimental methods will be outlined below.

*MLP Classifier*

Pending description.

Mlpc = MLPClassifier(hidden\_layer\_sizes=(20,20,20))

This model yielded 85.85% on the test set.

*MLP Regressor*

Pending description.

Mlpc = MLPRegressor(hidden\_layer\_sizes=(20,20,20))

This model yielded 85.77% on the test set.

*TensorFlow Sequential Neural Network*

Pending description.

Mlpk = Sequential()

Mlpk.add(Dense(20,activation=’relu’))

Mlpk.add(Dense(1,activation=’sigmoid’))

Mlpk.compile(loss=tf.keras.losses.binary\_crossentropy,optimizer=’sgd’,metrics=[‘accuracy’])

Mlpk.fit(X\_Train,Y\_Train,epochs=10)

This model yielded 77.38% on the test set.

*Almanac Column Code*

Pending description.

Time-Series Focused Preprocessing

The additional experimentation described in this section involved transforming the dataset in a manner that better captured time-series elements. The theory behind this experiment is that the recent time-series data would be able to offer insight as to whether a storm system is moving into or exiting a certain area, thus providing potentially useful features in predicting future rainfall.

This time-series transformation was executed by generating a pivot table on given date that produced the mean values of all features present in the data set for the previous *n* days, and then merging these newly generated features back in with the existing dataset. This preprocessing can be better visualized in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Location | Original Features | New Features |
| *d* | *L* | *F* | *F*avg between(*d-n,d*) |

All existing models were tested following execution of this transformation for all values of *n* from 1 to 7. The models run on the time-series transformed data produced similar accuracy to those run on the original dataset, indicating that this experiment did not offer any improvement. There was only minimal improvement observed while rerunning the ‘liblinear’ Logistic Regression model on the time-series infused data, producing an accuracy of 86.0, but the improvement was not significant.

# Discussion and Conclusions

Pending Discussion and conclusions