Tyler Anderson  
DSC 630   
Preliminary Analysis

**Australia Weather Prediction**

Abstract

In this project I try to answer the question that whether or not it will rain the next day in Australia. This project implements Logistic Regression and Scikit-Learn libraries. To answer this question, I built a classifier to predict whether or not it will rain the next day in Australia. I trained a binary classification model using logistic regression. I used a dataset that contains 10 years of daily weather observations from 2008-2017 from the Australian weather stations. Although the dataset is very complete there is some cleaning steps that need to be done prior to training the dataset and fitting it to our binary classification model. The dataset was retrieved from Kaggle and the description for the dataset suggested to remove RISK\_MM variable that may skew results so that was addressed in the cleaning steps. My target variable is RainTomorrow that gives a binary output of Yes or NO, this allows us to run our model over our entire dataset and get a good accuracy. I used feature scaling to normalize the range of independent variables and features of our train and test sets, this will follow with our model training of our training set to be fed into the Logistic Regression classifier. Once the train and test set were created, we could use sklearn for our Logistic Regression and fit it into a binary classification model. This Model gave us an accuracy score of 85%. Overall this is a good enough accuracy to use in the real world and to adopt for weather stations and may even be improved on in the future with more data.

Intro/Background of the problem

Weather stations in Australia have gathered weather data from 2008 to 2017 and want to be able o use that data to predict if it will rain the next day. This dataset contains daily weather observations from numerous Australian weather stations. The target variable RainTomorrow means: Did it rain the next day? Yes or No. The dataset description states that we must remove the RISK\_MM variable if you are using this dataset for predicting whether it will rain the next day. RISK\_MM is the amount of rainfall in millimeters for the next day. It includes all forms of precipitation that reach the ground, such as rain, drizzle, hail and snow. And it was the column that was used to determine whether or not it rained to create the binary target. For example, if RISK\_MM was greater than 0, then the RainTomorrow target variable is equal to Yes.

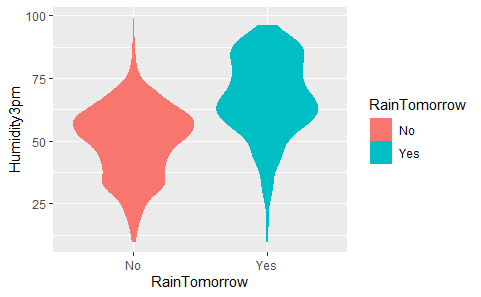
Since it contains information about the future, and since it contains information directly about the target variable, including it would leak the future information to the model. Using it as a predictor to build a model and then testing on this dataset would give the false appearance of a high accuracy. It is included in the dataset so that if you wanted to create your own binary target and decide that a small amount of rain like 0.1 mm shouldn't be counted as a rainstorm, you could try predicting only more significant amounts of rain.

It's also included in the dataset so that, if you wanted to, you could use the dataset to build a regression machine learning model -- instead of classification. In other words, you can use RISK\_MM as your target and drop RainTomorrow if you want to treat this as a regression problem instead of a classification problem. For this project however, we will be using the RainTomorrow variable for our target variable and drop RISK\_MM since we want to do a binary classification model.

Methods

I started my project with some exploring in R, I used the data.table, tidyverse and plotly packages since they contain many smaller packages within them and I don’t have to type out each package to load them. I started out with a histogram of rainfall using ggplot to show me the frequency of rainfall for each mm. This plot included days that did not rain so the histogram was skewed right with the majority at 0 mm. Because of this I wanted to see how many days it rained versus how many days it did not rain.

Overall it rained 22.4% of the time between 2008 to 2017. I wanted to take a quick look at the humidity levels at 3pm, the time where humidity would be highest, for each and to show the frequency, to do this I had to use geom\_violin to give me what I wanted and overall I could see that high humidity was a large factor in determining if it rained or not, however I did notice that there was a high amount of days it did not rain with a humidity between 50-75% with a peak at about 60% humidity that it did not rain. Humidity higher than 60% showed an increase in the likelihood of it raining.



I did the same thing with air pressure at 3pm. Using the same type of plot, we could see some differences but nothing too distinctive other than that the possibility of rain was true across atmospheric pressure (hpa) between 1010-1020 hpa. However, if it did not rain the pressure was most likely somewhere at 1017 hpa.

Most of my project was done in python where visualizations, cleaning and model building were done as well as logistic regression. Exploring the data, I could see there were 42193 instances and 24 variables in the data set. I viewed the columns to show me the variable names. In the decryption it recommends dropping the variable RISK\_MM if we are predicting rain the next day using Logistic Regression. I dropped RISK\_MM using *df.drop(['RISK\_MM'], axis=1, inplace=True)* and confirming the variable is dropped with *df.info()*. During this process I could see that there are categorical and numerical variables when noticing that some were listed as float64 or objects. Categorical variables listed as objects and numerical variables listed as float64. Using *df.describe()* allowed me to view the statistical properties of the numerical variables to see if there were any discrepancies in the data.

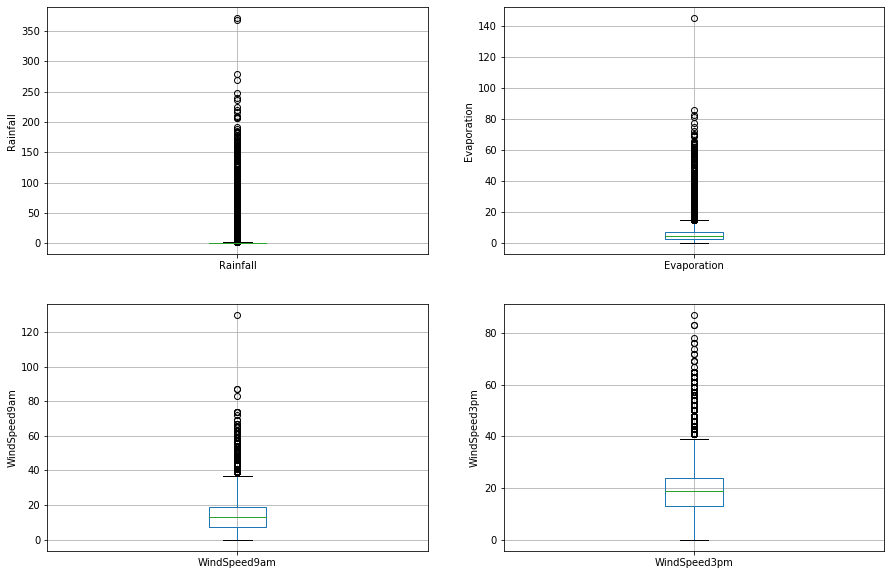
I started more of my analysis with some Univariate analysis. I used RainTomorrow as my target variable. I fist had to make sure that my target variable did not have any missing values and using *df['RainTomorrow'].isnull().sum()* confirmed there are 0 missing values. From our R script we could see that there are 2 unique variables that consist of ‘Yes’ and ‘No’ making out target variable binary. I could then see exactly how many days it did rain and how many days it did not rain. It rained a total number of 31,877 days and did not rain 110,316 times over the 2008 to 2017 period. Our Univariate analysis concluded that the RainTomorrow variable has 2 unique values. The two unique values are No and Yes. Out of the total number of RainTomorrow values, No appears 77.58% times and Yes appears 22.42% times. The ‘No’ variable have 110,316 entries, and the ‘Yes’ variable have 31,877 entries.

We can continue with our Bivariate Analysis. For out Bivariate Analysis I segregate the dataset into categorical and numerical variables. There is a mixture of categorical and numerical variables in the dataset. Categorical variables have data type object. Numerical variables have data type float64. I want to first explore the Categorical variables, those variables are 'Date', 'Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm', 'RainToday', and 'RainTomorrow'. Looking at this I notice date is considered a categorical variable that I will have to break up into smaller numerical variables later.

Checking for missing values we can see that there are only 4 categorical variables in the dataset which contains missing values. These are WindGustDir, WindDir9am, WindDir3pm and RainToday. Next I look up the labels for each variable to check for cardinality, the number of labels within a categorical variable. A high number of labels within a variable is known as high cardinality. High cardinality may pose some serious problems in the machine learning model. When checking for categorical variables I can see that date contains 3,436 labels while the next highest is location which has 49, meaning the Date variable has high cardinality. We can see that the Date variable needs to be preprocessed, all the other variables contain relatively smaller number of variables.

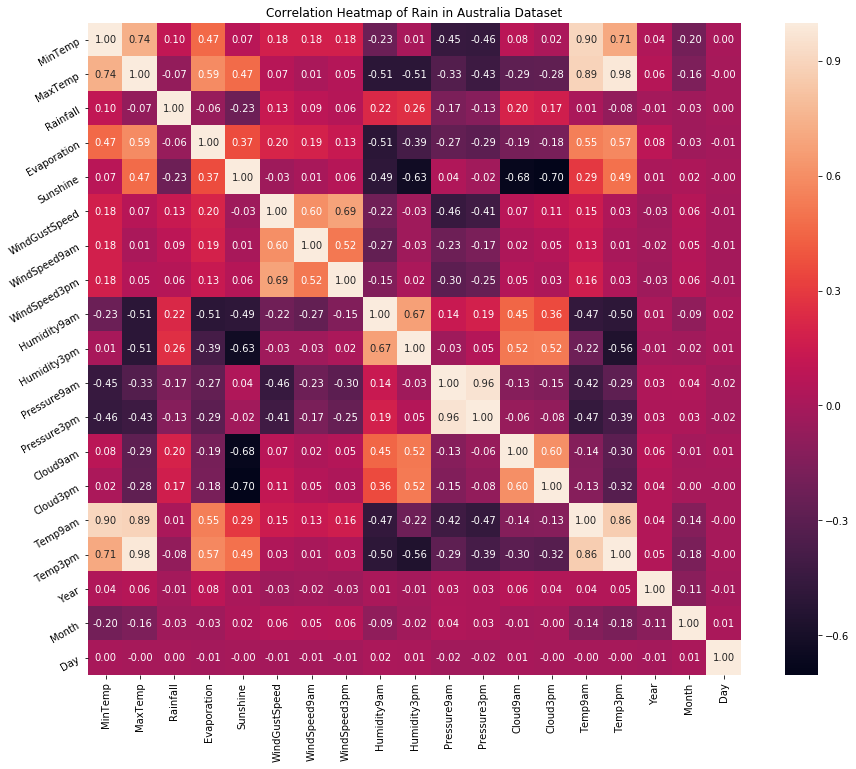
For the date variable we need to break it up using some feature engineering. I can parse the dates currently coded as strings into a datetime format using this string: *df['Date'] = pd.to\_datetime(df['Date'])* I can then continue to extract the year, month and day from the Date variable and create new columns created from the date variable. I can then remove the date variable from the dataset *df.drop('Date', axis=1, inplace = True).* Previewing the data again I can see its been removed.

I can now move on to exploring the numerical variables. There is a total of 19 numerical variables, MinTemp, MaxTemp, Rainfall, Evaporation, Sunshine, WindGustSpeed, WindSpeed9am, WindSpeed3pm, Humidity9am, Humidity3pm, Pressure9am, Pressure3pm, Cloud9am, Cloud3pm, Temp9am and Temp3pm. All the numerical variables are of continuous type. The numerical variables will have to be checked for missing values and outliers. By printing this code *df[numerical].isnull().sum()* I can see that all numerical variables contain missing values. Veiwing summary statistics with *print(round(df[numerical].describe()),2)* we can see that the Rainfall, Evaporation, WindSpeed9am and WindSpeed3pm columns may contain outliers. To visually see these outliers, I can plot boxplots to visualize the outliers.

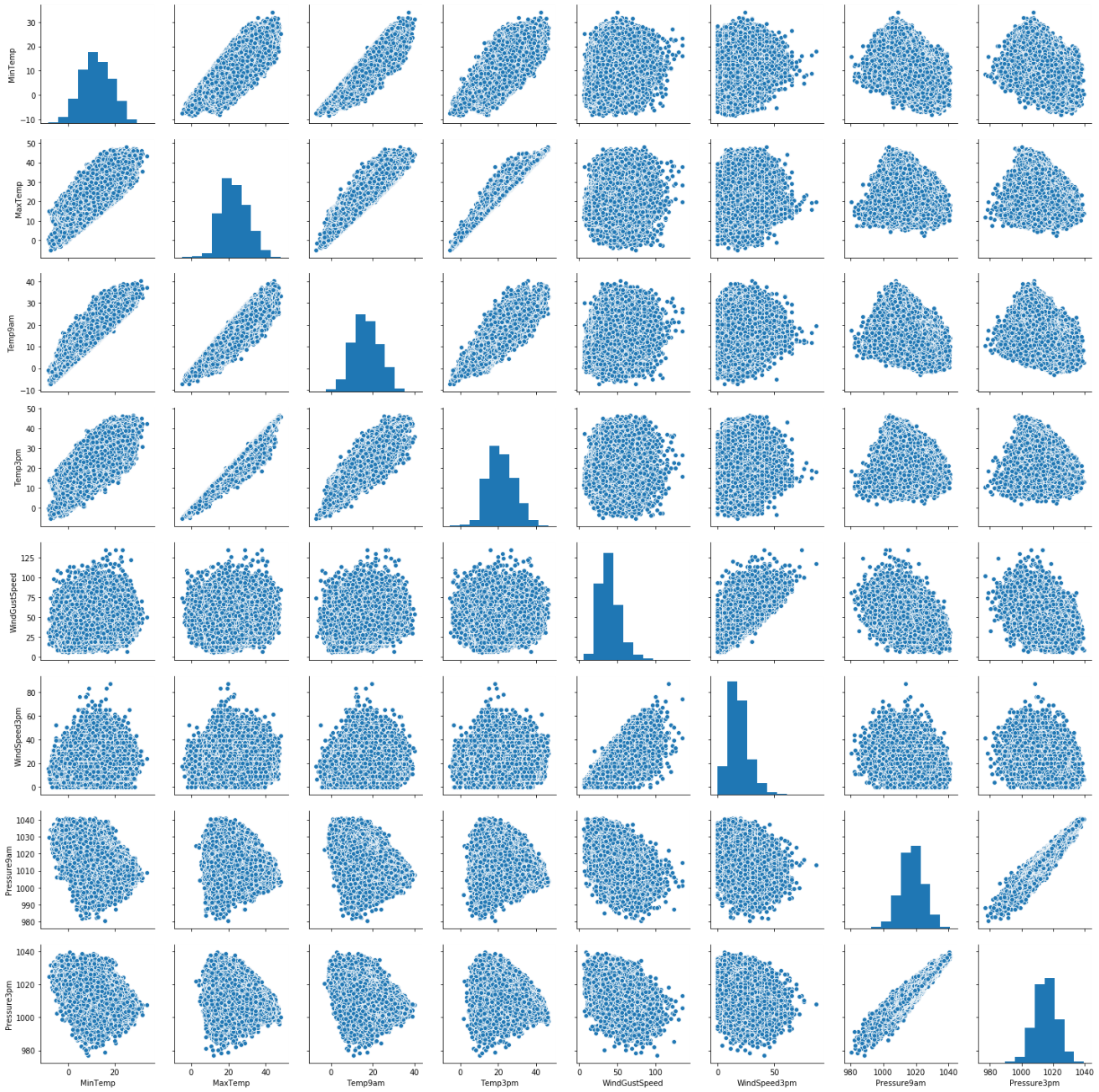


I can now plot histograms to check distributions to find out if they are normal or skewed. Plotting them as box plots checks the distributions to see if they are normal of skewed. My results indicate that they are skewed to the right. Because they are skewed, I can find the IQR or Interquantile range. The interquartile range (IQR) is the difference between the first quartile and third quartile. Since the variables are skewed IQR is my best choice to find outliers. Using help from Stack Exchange and Kaggle my results for IQR are:

* Rainfall outliers are values < -2.4000000000000004 or > 3.2
  + For Rainfall, the minimum and maximum values are 0.0 and 371.0. So, the outliers are values > 3.2.
* Evaporation outliers are values < -11.800000000000002 or > 21.800000000000004
  + For Evaporation, the minimum and maximum values are 0.0 and 145.0. So, the outliers are values > 21.8.
* WindSpeed9am outliers are values < -29.0 or > 55.0
  + For WindSpeed9am, the minimum and maximum values are 0.0 and 130.0. So, the outliers are values > 55.0.
* WindSpeed3pm outliers are values < -20.0 or > 57.0
  + For WindSpeed3pm, the minimum and maximum values are 0.0 and 87.0. So, the outliers are values > 57.0.

I can now move on to Multivariate Analysis. In this step I want to discover patterns and relationships between variables in the dataset. Drawing a heatmap will give me correlations between the variables.

We can see that 'MinTemp', 'MaxTemp', 'Temp9am', 'Temp3pm', 'WindGustSpeed', 'WindSpeed3pm', 'Pressure9am’ and 'Pressure3pm are all highly correlated with one another. Using the highly correlated variables I can extract them and draw a pair plot to look at the direct relationship between them.



Now I can declare a feature vector for the target variable RainTomorrow and split the data into separate training and testing sets using the sklearn package. My train and test variables will be listed as X\_train and X\_test. I can now do some more feature engineering particularly with transforming the data into something more useful for the categorical and numerical variables. I first need to look at both separately to see their missing values and to start to remove them. I am not sure if there is a pattern of the missing data or things are left out purposefully and it may be that the missing values are completely random and not really be indicative of something else. There are multiple ways to remove the missing data, but I can impute them since there are outliers in the data. Ill use imputation on the training and test sets to try to avoid overfitting. I can do this for both numerical and categorical values.

I can start with imputing missing values in X\_train and X\_test with respective column median in X\_train. After removing the missing values I can check for missing values again with *X\_train[numerical].isnull().sum()* and it shows 0 missing values for all numerical variables. I can do the same steps with the categorical variables and test both the train and test sets for missing values after I impute all categorical and numerical variables for missing values.

I can now move on to the outliers in the train and test sets. We have seen that the Rainfall, Evaporation, WindSpeed9am and WindSpeed3pm columns contain outliers. I can use top-coding approach to cap maximum values and remove outliers from the above variables. As an example of the process I can use this block of code to execute the top-coding approach.

def max\_value(df3, variable, top):

return np.where(df3[variable]>top, top, df3[variable])

for df3 **in** [X\_train, X\_test]:

df3['Rainfall'] = max\_value(df3, 'Rainfall', 3.2)

df3['Evaporation'] = max\_value(df3, 'Evaporation', 21.8)

df3['WindSpeed9am'] = max\_value(df3, 'WindSpeed9am', 55)

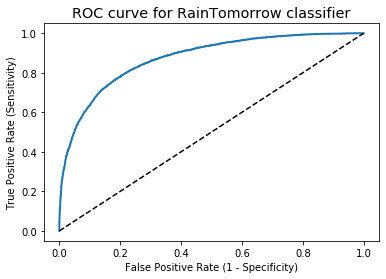
df3['WindSpeed3pm'] = max\_value(df3, 'WindSpeed3pm', 57)

This allows me to look at each variable and look at the maximum value for each with *X\_train.Rainfall.max(), X\_test.Rainfall.max()* as an example of both train and test sets to make sure both values are the same. This also caps the maximum values for all the numerical values to get rid of the outliers. We can now encode the categorical variables; I want to focus on the RainToday variable. When we encode the RainToday variables we can see that two additional variables RainToday\_0 and RainToday\_1 are created from RainToday variable. Once this is done I can the X\_test testing set and the X-train training set, after completing this we can use the train and test sets for model building but first I should map all feature variables onto the same scale called feature scaling.

I can map all the feature variables by describing the X\_train data to view the dataframe. I can create a variable called ‘cols’ by passing through the columns from X\_train through it. Using sklearn MinMaxScaler I can get the X\_train dataset ready to be fed into the logistic Regression classifier. I train the logistic regression model on the training set using sklearn LogisticRegression and instantiate the model *logreg = LogisticRegression(solver='liblinear', random\_state=0)* and fit it to the model *logreg.fit(X\_train, y\_train).* We can now predict our results.

Results

To predict the results, I use the predict\_proba method that can givce me the probabilities for the target variable (0 and 1) in an array form. In this case 0 is for probability of no rain and 1 is for probability of rain. For my probability of getting output 0 for no rain I use *logreg.predict\_proba(X\_test)[:,0]* for probability of getting output 1 for rain I use *logreg.predict\_proba(X\_test)[:,1].*

To check my accuracy, I start with my test set. I can use sklearn metrics accuracy\_score package and run this script *print('Model accuracy score: {0:0.4f}'. format(accuracy\_score(y\_test, y\_pred\_test)))* I get an accuracy score of 0.8501. Here, y\_test are the true class labels and y\_pred\_test are the predicted class labels in the test-set. I can now do the same thing for my train set and get an accuracy score of 0.8476. The training-set accuracy score is 0.8476 while the test-set accuracy to be 0.8501. These two values are quite comparable. So, there is no question of overfitting. Lastly, we can determine how well out model does its prediction by plotting a ROC curve. This curve indicates the model does a good job of predicting how well our model predicts the rain tomorrow in Australia

Discussion/Conclusion

Overall our logistic regression model accuracy score was 0.85. So, in short, the model does a pretty good job of predicting whether it will rain or not in Australia. Of all the observations the majority will predict that there will be no rain tomorrow. The model also shows no signs of overfitting. My ROC curve was well bowed or approached 1 so it did a very good job in predicting if it will rain in Australia tomorrow or not. A couple of things have occurred to me during this project and its conclusion. I wondered how the accuracy change would if it used the data from multiple days before to predict rain. So, instead of the previous day predicting the rain the next day what about the previous 2 or 3 days? Would the accuracy change any? Would the accuracy be better or worse? With the current accuracy we got for the project I would say it can definitely be used with new data to try to determine rain in Australia in a live scenario.

Acknowledgments

Thank you to Bellevue University and Catie Williams for hosting this course. I learned a lot doing this project and built my skills even more that I thought I would. There were a lot of bumps in the way but struggling through the project has helped me learn more that I ever thought I would.

References

# Works Cited

Abbott, D. (2014). *Applied Predictive Analytics.* John Wiley & Sons.

howtobuildsoftware.com. (2020). *How should the interquartile range be calculated in Python?* Retrieved from howtobuildsoftware.com: https://www.howtobuildsoftware.com/index.php/how-do/b9l0/python-statistics-median-percentile-wolframalpha-how-should-the-interquartile-range-be-calculated-in-python

Menon, A. (2019, Dec 17). *Logistic Regression in Machine Learning using Python*. Retrieved from towardsdatascience: https://towardsdatascience.com/logistic-regression-explained-and-implemented-in-python-880955306060

Ranjith, S. (2020). *predict\_proba for classification problem in Python*. Retrieved from CodeSpeedy: https://www.codespeedy.com/predict\_proba-for-classification-problem-in-python/

Singham, L. (2017, Jul 12). *Binary Classification in Python - Who's Going to Leave Next?* Retrieved from lukesingham.com: https://lukesingham.com/whos-going-to-leave-next/

Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media.

Müller, A. C., & Guido, S. (2016). Introduction to machine learning with Python: a guide for data scientists. " O'Reilly Media, Inc.".

Siegel, E. (2013). Predictive analytics: The power to predict who will click, buy, lie, or die. John Wiley & Sons.

https://www.kaggle.com/jsphyg/weather-dataset-rattle-package