Author: Ze Hong Wu

Embedded links in the descriptions of Existing Work from other people lead to cells in their Kaggle notebooks where they implement the described task.

Milestone 2, DS 675 Project

Dataset Description

For the purposes of this Milestone, we will focus on a dataset of Australian weather data from 2007 and 2017. The data is sourced from the Australian Bureau of Meteorology. The dataset is available here: https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package .

Each entry in this data corresponds to weather measurements for a given day at one of many different locations across Australia. The columns in the data set describe minimum and maximum temperatures (in degrees Celsius), rainfall for the day (in millimeters), hours of sun, as well as a number of measurements taken at 9am and 3pm local time: direction and intensity (km/h) of wind, humidity (%), air pressure (hectopascal, equivalent to millibar), and temperature. Note that the 9am and 3pm temperatures are distinct from the min and max temps. The wind direction is categorical, while all other relevant columns are numerical. The data set has two additional categorical columns that can be used for classification: whether it rained on the given day and whether it rained on the following day, both expressed as "Yes" or "No".

For this data set I plan to carry out a classification task. The data set authors suggest using current-day weather data to predict whether there will be rain on the next day. I can also try to predict whether there was rain on the given day using the weather data columns, though this task has a smaller real life relevance. My only concern is that, since different places in Australia have different climate and weather behavior that are not reflected in this data set, this may cause the machine learning model I choose for this data set to perform suboptimally.

Existing Work 1

The most notable instance of existing predictive learning work that previous people carried out on this data set is a tutorial on building a logistic regression classifier on this data set. This tutorial also happens to have the most upvotes out of all Kaggle notebooks for this data set. The code is available here:
https://www.kaggle.com/code/prashant111/logistic-regression-classifier-tutorial .

The predictive model this author used was scikit-learn’s Logistic Regression model. The author carried out a standard procedure of loading the dataset, carrying out exploratory data analysis to understand what is happening in each of the features, identifying a y-feature (in this case, RainTomorrow), engineering features using one-hot encoding and filling in of missing values, creating train-test splits, training the model, and reflecting on the results. The training step was not exceptional, with the author using the model class built-in instantiate and fit functions.

However, I found much of the work leading up to the model training as well as some of the concluding review to be both very informative for my future work and educational. More specifically, the author went into depth on preparing and cleaning the data set to a training-ready state.

During my previous machine and deep learning studies I learned a common saying in the industry: garbage in, garbage out. No model can perform well if it is trained on bad data. The Australian weather data set contains a number of flaws in it, which the author of this code explains and handles.

The data set contained entries that had missing values in some features, which generally occurs in real-life data sets due to the effects of human error on data collection. The author replaced missing categorical features with the most common value for each feature and then one-hot encoding the features in question. I feel that this may be a subpar choice, since this process may assign incorrect features (such as marking a row's RainToday value as Yes when other columns with similar features have RainToday values of No). The author also previously proposed one-hot encoding categorical features with one additional one-hot column to denote a missing value. I am more in favor of this option since it does not directly create contradicting inputs as described above.

For missing numerical features the author carried out statistical analysis on them to determine their distribution and found that the numerical features were generally normally distributed with some outliers. Using this information the author chose to impute the missing numerical features with the median of existing values. This statistical analysis has much more in common with the coursework from my Applied Statistics course than with what we learned in DS675 so far. The author concluded by normalizing the numerical data using scikit-learn's min-max scaler.

After training the model and evaluating its performance against the testing set the author also reflected on their model's behavior. The author tested different regularization strengths, checked for overfitting by comparing training and testing set results, and also checked for the null accuracy by comparing their model against a "null prediction" of always guessing the most common class. While we did learn about regularization and overfitting/underfitting in class, we did not discuss null accuracy in the context of categorical prediction.

Overall, this Kaggle notebook provided me with a good refresher on pre-training tidying work that I learned in previous coursework as well as a standard data science process/structure that I can reference for my project.

Existing Work 2

Another instance of other people's predictive classification work on this data set takes a similar format as the first Existing Work. However, instead of using a logistical classifier, this one uses an artificial neural network. The code is available here:

https://www.kaggle.com/code/karnikakapoor/rain-prediction-ann ; note that the Table of Contents on the right side sometimes does not function properly.

This author carries out the same data science process as the author of the previous Existing Work by preprocessing and engineering the data set before splitting it into train-test sets and training a model on them. However, besides the choice of model, this author also chose different engineering and cleaning methods.

Instead of one-hot encoding the categorical features, the author chose to use scikit-learn's label encoder to convert categories into numerical values (from 0 to the number of classes - 1). I have my concerns regarding this method since it creates the illusion that categories assigned to higher numerical values are "greater" than categories assigned to lower values. For example, a wind direction of "NE" being assigned to a bigger number than "SE" was assigned to implies that a northeasterly wind is somehow greater than a southeasterly wind.

Instead of accepting outliers as the previous example did, the author chose to discard all rows with outlier numerical features. On one hand, the remaining rows are less likely to stand out from the main yes/no prediction clusters. On the other hand, this discards training data that might help train the model against a wider variety of scenarios.

For the predictive model the author chose a simple artificial neural network, constructed using the Keras Sequential class. The model in question contains several dense layers of various unit counts using ReLU activation, connection dropout layers near the later half of the model, and a single sigmoid unit as the output layer to predict a categorical value. Additionally, the author used an early stopping callback to halt learning before the model's learning became arbitrarily slow. Our coursework indicates that we will learn about artificial networks during the first week of April, so this is content we have not yet covered.

The author concludes their work by evaluating their model's performance. When evaluated against the test set, the ANN achieved an accuracy of 85%. This performance is comparable to the 85.01% accuracy score the logistic regression model from Existing Work 1 achieved, despite the increase in predictive power a dense ANN provides over a single logistic neuron.

Opinions and Reflections

During my review of these two Existing Works I changed some of my pre-existing assumptions about how I will proceed with my project. During Milestone 1 I proposed using "RainToday" as the predicted feature instead of "RainTomorrow". However, Existing Work 1 showed that many rows lack values for that feature. If I were to pursue that option, I will have to account for this by discarding rows with missing values. Fortunately, since there are only around 1400 missing values for "RainToday" (compared to around 145,000 overall rows), this should not be a major loss. I am opposed to imputing nulls with the more common category since that would effectively be me making up labels to predict for.

During my discussion on the two Existing Works I commented on some of the choices the respective code authors made and whether I agreed with them or not. I plan on applying these same conclusions to my future Milestones.

Due to the way the data set is structured, “RainToday” and “RainTomorrow” are the two most obvious targets to predict for. Alternatively, instead of doing classification prediction on either of these features, I could carry out regression prediction on “Rainfall” (which describes the amount of rain on that given day in mm). This approach differs from those of the most upvoted Kaggle notebooks for this data set, almost all of whom focused on data visualization or predicting “RainTomorrow”. This would also require a different approach compared to either of the Existing Works due to the different output format which affects everything from activation functions to error calculation to accuracy evaluation.