جامعة عجمان
AJMAN UNIVERSITY

College: Engineering and Information Technology
Department: Information Technology
Program: Data Analytics

Introduction to Programming 2 Course Project

Rain Prediction using ANN

Prepared by:
202010914 Maryam Almetnawy
202020278 Raghad Yousef
202010959 Reem Abu Hasna

Supervised by:
Dr. Salam Fraihat

Academic Year 2022 - 2023 – Fall

Rain Prediction using ANN

Maryam Almetnawy
Data Analytics
202010914@ajmanuni.ac.ae
Ajman University

Raghad Yousef
Data Analytics
202020278@ajmanuni.ac.ae
Ajman University

Reem Abu Hasna
Data Analytics
202010959@ajmanuni.ac.ae
Ajman University

***Abstract*— The purpose of this research is to implement a Neural network model using Keras to predict rainfall. In this paper, we have discussed the problem that farmers face when planting crops during rainy seasons, as well as, the ANN architecture and its brief background, the experiments and methodologies involved in the process, a comparison between our implemented ANN model & other related work. By the end of this research, we aim to have a thorough understanding and analysis of how the ANN model is implemented to solve a real-world problem.**

***Keywords* — Rainfall Prediction, Keras, Deep Learning, Artificial Neural Networks, Random Forest.**

I. Introduction

Making daily business decisions for a farmer might be aided by weather forecasts. These choices include when to fertilize, when to irrigate crops, and which days are best for fieldwork.

A farmer must be mindful of the moisture, light, and temperature in order to grow a successful crop. It is necessary to have access to comprehensive weather data, including historical records, current weather, and future forecasts. Although most people view rain as a good thing for fields and crops, most crops have a "optimal" quantity of rainfall during any given growth season. [1]

For predicting rainfall, a variety of methodologies are available, including machine learning methods, statistical methods, and numerical weather prediction (NWP) models. Due to the highly complicated and non-linear physical processes determining rainfall occurrence, machine learning techniques including artificial neural networks (ANN), k-nearest neighbors, and random forest models are more suitable for rainfall forecasting. [2]

Therefore, based on the neural network prediction model, we have developed a method that can assist farmers in their decisions. Farmers would be able to decide when to plant and when to use the available water in the farm to water the crop till rain comes!

II. Task and Data

A. Data Description

The dataset used in the project is generated from an Australian daily weather government site, where its observations were drawn from multiple weather stations, that is then obtained from Kaggle. [3]
The data consists of daily weather observations from many locations across Australia from the year 2007 – 2017 and a sample of 145,460 data with information about 24 different types of attributes (numerical and categorical).

There are 2 different Boolean attributes "Rain Today & Rain Tomorrow" indicating whether it rains or not, which will be our classified target that will be tested to predict rainfall tomorrow using Artificial Neural Networks.

B. Data Pre-Processing

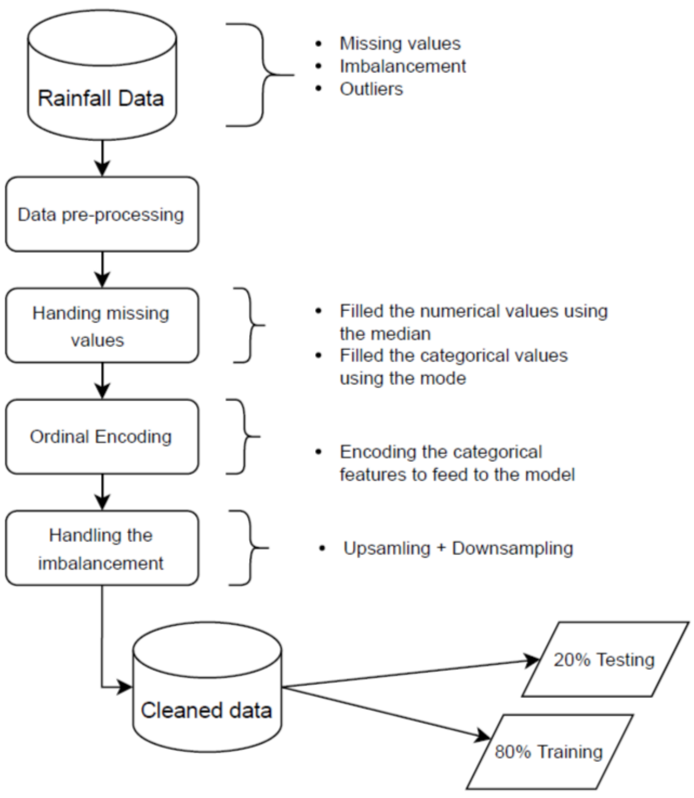

Rainfall Data
Missing values
Imbalancement
Outliers
Data pre-processing
Handing missing values
Filled the numerical values using the median
Filled the categorical values using the mode
Ordinal Encoding
Encoding the categorical features to feed to the model
Handling the imbalancement
Upsamling + Downsampling
Cleaned data
20% Testing
80% Training

Fig.1. Data Pre-Processing Summary

During the exploration of the data, we've came across a huge number of incomplete & missing values, which we have imputed using the central tendency measures, since skewed data works the best with mean/mode imputations [4] . For the categorical attributes, we've filled the missing values using the mode while filled the null numerical features with the mean.
In the target attribute "Rain Tomorrow", we noticed a huge imbalancement that needed to be dealt with as well, which can be seen in figure.2.

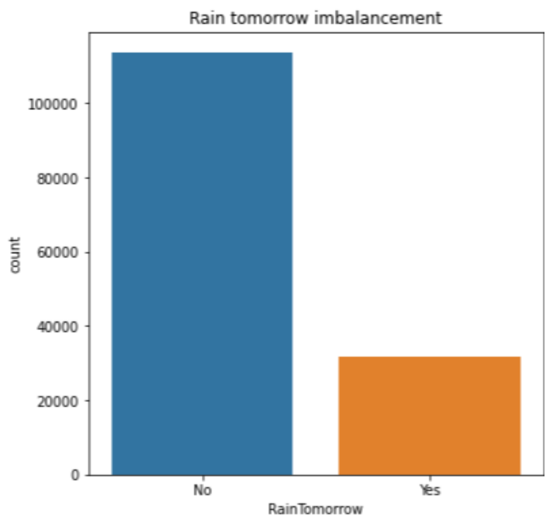


Fig.2. Imbalancement in target feature "Rain Tomorrow"

After normalizing the data, we balanced it using both the up-sampling and down-sampling approaches. This is done because a small bit of oversampling for the minority class samples enhances its bias, and a small amount of under sampling for the majority class samples reduces its bias. Although our values are not exactly equal, they are better balanced as a result. The attribute "Rain Tomorrow" is shown in Figure.3 after the process.

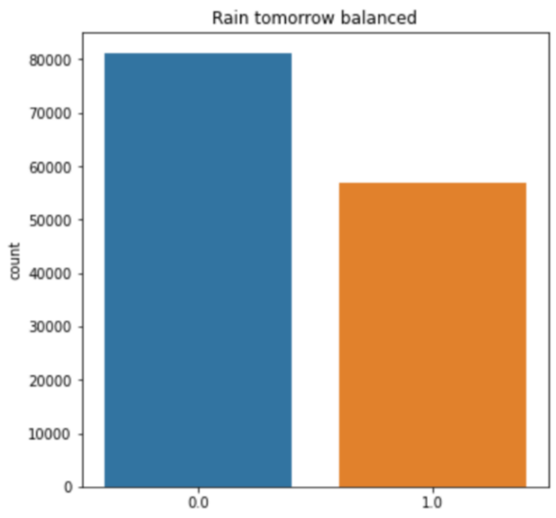


Fig.3. Balanced target feature "Rain Tomorrow"

Since the target features were of type Boolean, Ordinal Encoding was applied to encode the categorical features in the data so it can be easily feeded into the model. To finalize our pre-processing, figure.4. shows the number of noticeable outliers we have found, but we didn't handle them before standardizing the data, since having values that are highly different than usual makes sense. Some days would have normal predictions of rainfall while other days might have extremely high predicted rainfalls.

And by that, our data is cleaned and ready to apply Machine learning / Deep learning models on.

Figure.1. shows a summary of the data cleaning process.

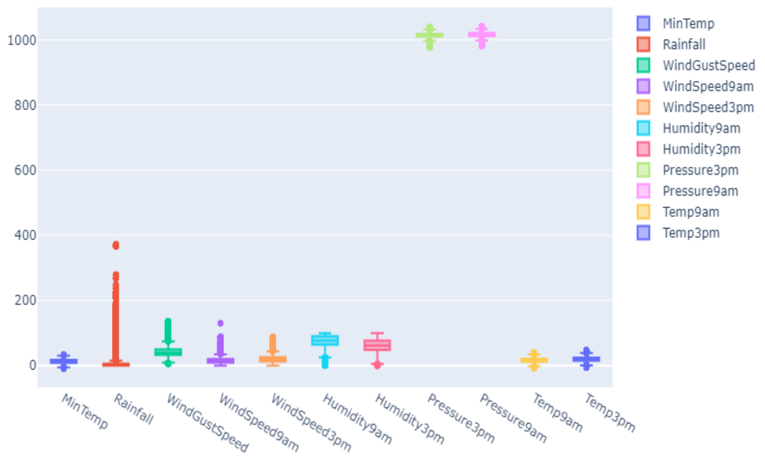


Fig.4. Outliers Box Plot

C. Evaluation

The Artificial Neural Network model was evaluated using the Classification Report since our output was either if it will rain today or not. The report displays the precision, recall, F1, accuracy and support scores for the model.
Four different metrics were used to evaluate the implemented model:

- Precision – How often predictions were actually correct?
- Recall - What percent of the positive cases did we capture?
- F1 score – What percent of the positive predictions were correct? [5]

Check figure.5. for evaluation metrics equations.
We evaluated the accuracy and loss of the model using py-plots too, so we can see if there is any overfitting or underfitting in the data, according to that we added a drop out layer to avoid the overfitting we got.

$$P = \frac{TP}{TP + FP}, \qquad R = \frac{TP}{TP + FN},$$
$$ACC = \frac{TP}{TP + FP + TN + FN}, \qquad F_1 = 2 * \frac{P * R}{P + R}$$

Figure.5. Evaluation metrics equations

III. Methadology

A. The Artificial Neural Netowrk

The ANN is a computational model that replicates how brain nerve cells function. As the fundamental unit of information processing, brain nerve cells make up the majority of the human nervous system. [6]
The fundamental element of information processing in the ANN idea is referred to as a neuron. Backpropagation is one of the numerous forms and applications of the training process for ANNs. In order to produce more accurate prediction results with a minimum amount of error, this study suggests an ANN algorithm to forecast rainfall data by studying and analyzing the patterns of the historical data.

B. Structure

The current research focusing on predicting rain to help farmers detect rain helping them in their planting decisions, since it's a worldwide problem. Data was yielded from IOT weather sensors in Australia [3] that was then obtained from Kaggle by us, to apply EDA, visualizations, and the ML / ANN models on. Figure.8 shows the methodology flow chart summary.

Some of the visualizations we have implemented focused on exploring the data to help us understand which features correlate more with having rain on the next day. Since time series data is very sensitive to work with and due to it not having quality, leads to prediction failures, according to the findings we have found, we chose to work with a classification algorithm and not a regression problem.

During our EDA process, we have seen a lot of low data quality that was concerning, since it will affect feeding it into the ANN model.

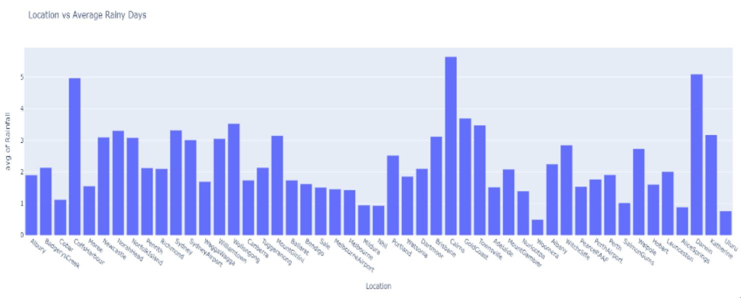


Fig.6. Locations vs Average Rainy Days Plot

With the help of the tools that are offered for visualizations we were able to implement figure.6, to see how high rainfalls are at different locations and if locations have any impact on daily rain. According to the plot, some locations experience more rainy days than others, Darwin city gets an average of 6% rainfall daily while Woomera on the other hand gets only 1% of rainfall daily! After aggregating the date attribute into months, days, and years, we were also able to see the likelihood of rain across the years, shown in figure.7, the time series plotted against the Maximum Temperature, since it was one of the attributes that had high correlation with most of the features.

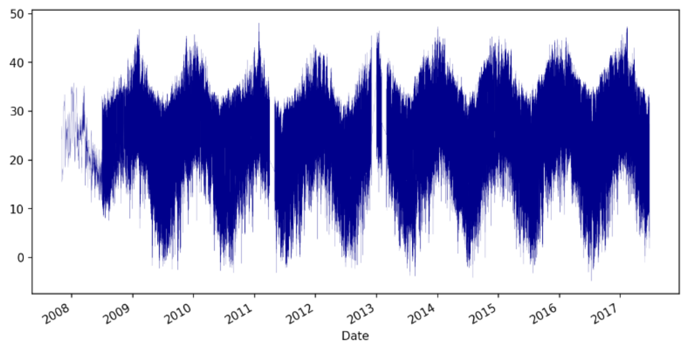


Fig.7. Maximum Temperature vs Year in Australia

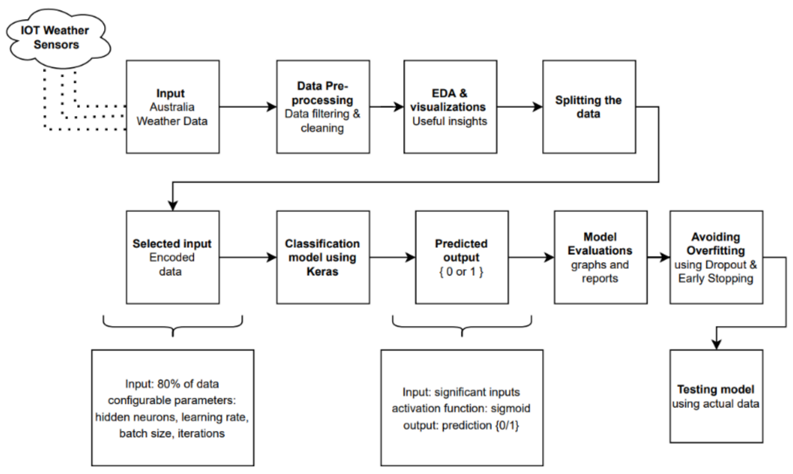
By visualizing this graph, we were able to understand that the likelihood of having a rainy day is higher if the temperature difference between the minimum and maximum temperatures is small.

Data pre-processing was done to the data to remove any difficulties in feeding into the model using pandas, NumPy, random over sampler and random under sampler packets.

Splitting the data to 80% training and 20% testing was done during feature selection process after encoding the values using ordinal encoding.

The input was fed into the classification Keras sequential model using the Dense layers, which are fully connected and multiplied by the random weights that determines the importance of each neuron, that then goes through the ReLU activation function and the hidden layers. While it gives us the predicted binary output, either zero or one, rains tomorrow or not.

The output is then evaluated using graphs, plotting the training and validation accuracies, loss, confusion matrix and classification reports. While evaluating the model, overfitting was occurred in the first two experiments, so to help avoid this issue we used regularization in the third experiment, using the Dropout layer and an early stopping that stops the training when there is an overfitting occurred. Weight initializers were used to test the model accuracies among the 3 architectures, we started with random weights, then implemented it using the uniform weights, because they work the best with binary classification models.



IOT Weather Sensors
Input Australia Weather Data
Data Pre-processing Data filtering & cleaning
EDA & visualizations Useful insights
Splitting the data
Selected input Encoded data
Classification model using Keras
Predicted output { 0 or 1 }
Model Evaluations graphs and reports
Avoiding Overfitting using Dropout & Early Stopping
Input: 80% of data configurable parameters: hidden neurons, learning rate, batch size, iterations
Input: significant inputs activation function: sigmoid output: prediction {0/1}
Testing model using actual data

Fig.8. Methodology Flow Chart

IV. Experiments

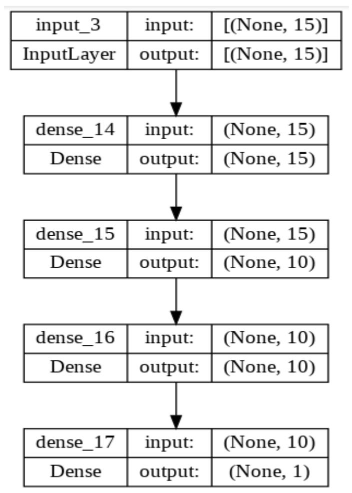
A. Motivation

The basic motivation in this research is to develop a fast and new efficient method for classifying rainfall predictions using Keras. By feeding the data into a neural network model we were able to achieve and learn new techniques in achieving better accuracy, avoiding overfitting, as well as, getting a higher performance model. During the experiments, we stopped refining the deep learning model and started optimizing the dataset instead by studying it more, cleaning it up, and understanding the nature of our dataset and its flaws. This boosts performance far more than tuning the model.

B. Description

Going through a full breakdown on our analysis, the dataset was firstly split into 80% training and 20% testing.
We tried 3 different experiments until we got an accuracy of 80% and a loss of 45% without any overfitting in the model.

Training model 1 consisted of all input features except for aggregated features during the EDA process, location, target features and some low correlated features.
Data was standardized to re-scale the features values with the distribution value between 0 and 1 that is useful for the optimization algorithms. The model was created using the Keras Functional API to have more flexibility over the layers.


input_3 | input: | [(None, 15)]
InputLayer | output: | [(None, 15)]
dense_14 | input: | (None, 15)
Dense | output: | (None, 15)
dense_15 | input: | (None, 15)
Dense | output: | (None, 10)
dense_16 | input: | (None, 10)
Dense | output: | (None, 10)
dense_17 | input: | (None, 10)
Dense | output: | (None, 1)
Fig.9. Model 1 Summary

During our first experiment, the input layer consisted of only 15 features/neurons, two hidden layers were added with 10 neurons each, and a ReLU activation function was used on the input & hidden layers, since we had only one output layer with one neuron, we used the activation function sigmoid on the output layer.
Dealing with a Binary classification problem, Cross Entropy as the loss function and Adam optimizer were used because the algorithm is for stochastic gradient descent. We fit the first model using only 20 epochs and 250 as the batch size. Which resulted in getting the following evaluations:

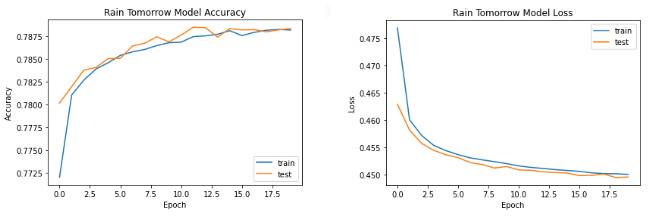
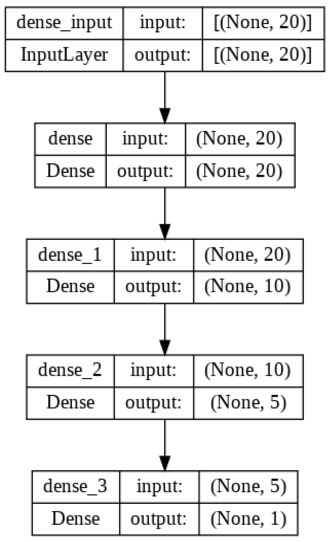


Fig.10. Model 1 Evaluations

Results aren't very accurate and having 20 epochs isn't enough to compile the model with, as well as it's seen that the network is having trouble fitting the training data. On the other hand, the test loss being lower than the train loss means there is a sampling bias which occurs when the training data is not sampled randomly enough from the collected data. To solve this issue, we had to reduce the bias by offering additional features.

By improving the issues, we had in the previous architecture; the number of features were increased in the input layer being 20 and tuned minor changes in the hyper parameters, figure.10. shows a summary of the updated model.


dense_input | input: | [(None, 20)]
InputLayer | output: | [(None, 20)]
dense | input: | (None, 20)
Dense | output: | (None, 20)
dense_1 | input: | (None, 20)
Dense | output: | (None, 10)
dense_2 | input: | (None, 10)
Dense | output: | (None, 5)
dense_3 | input: | (None, 5)
Dense | output: | (None, 1)
Fig.11. Model 2 Summary

Data-Preprocessing steps were changed during the second experiment, like handling missing values using the mean to median, handling outliers, and the approach of handling the imbalancement.
From using Keras Functional API to Sequential, we added the input layer with 20 neurons, adding the location of the city and returned the dropped attributes back to the features. Two hidden layers were created with 2 different neuron numbers, one consisting of 10, while the other had 5 neurons. Same activation functions were used but added a uniform weight initializer which makes the network performs effectively having sigmoid as an activation function for the output.
Earl stopping strategy was added to avoid any overfitting we have faced in the previous experiment and a goal to minimize the loss.
During the compilation of model 2, we added a learning rate which compromised of 0.0001 which is a hyper-parameter used to govern the way of an algorithm that updates or learns the values of a parameter estimate. So that means, increasing the number of training epochs is necessary.

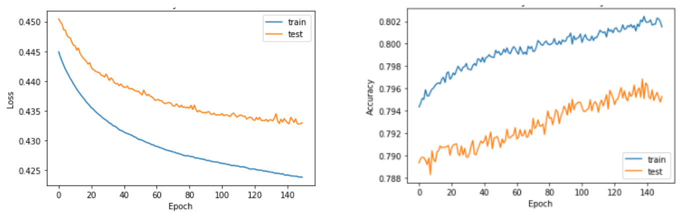


Fig.12. Model 2 Evaluations

Plotting the evaluations of the model helped us in seeing the performance of the model across each epoch clearly; Although we used early stopping and added some parameters and making the number of epochs increase to 150 has increased the overfitting between the training and validation data. As seen in Figure 12, having the training error lower than the test error indicates that the model has spent too much time learning the training data.

In order to solve this issue, we added a dropout layer(0.4) that means 40% of the training data is set to zero, but in testing data all features are used without freezing them.

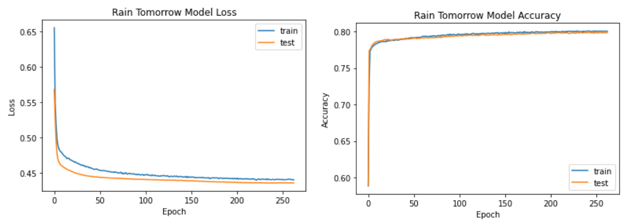


Fig.13. Model 3 Evaluations

Adding the Dropout layer is a technique where randomly selected neurons are ignored during training. So, 40% of the data was ignored making the model's overfitting issue disappear, as seen in figure 13, the gap between the training and validation accuracies has been steadily dropped during training and validation before stabilizing, this implies that the metrics of the model are probably great! Therefore, our accuracy has also been increased to 80%.

After some feedback we got from a trusted source. We tried pre-processing the data again, by dropping two features "Temperature 9am, Temperature 3pm", we also normalized the data instead of standardizing it because some features ranges were so high compared to other features. So, a total of 18 features were as input, early stopping was done with a patience of 100 and a total of 400 epochs where weights were initialized randomly, 2 hidden layers were added with a learning rate of 0.1 and an Adam optimizer as usual. The model stopped learning at 131/400 epochs where the execution time was 780 seconds.

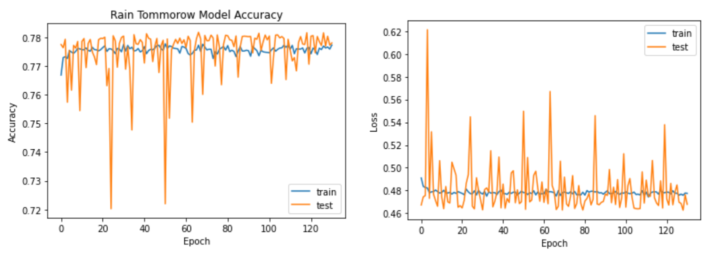


Fig.14. Model 4 Evaluations

For some reason, as seen in Figure14, the output of the model was a bit unpredictable, and different to the other models where they were more linear.
Due to that, we decided to add a drop out layer for our last experiment, changed the number of neurons and hidden layers. As well as used the "Tanh" as an activation function, to see if it makes any difference. The results were the best we got so far, shown in figure.15.

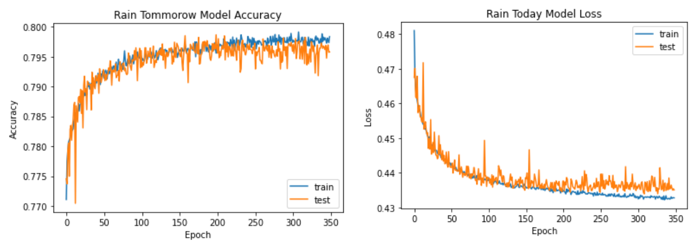


Figure.15. Model 5 Evaluations

C. Results

All three architectures were trained and assessed using the same dataset. Model 3 performed with a training accuracy of 80% and the quickest execution time. It had two hidden layers and a drop out layer. The training accuracy of the second model, which included two hidden layers, was 79%, 1% lower than that of the third model.
However, the model 3 architecture had a loss score that was 0.15% higher. The comparison of three distinct architectures is shown in Table 1. In comparison to the other two, architecture 3 is the most accurate. The evaluation leads to the conclusion that the suggested architecture delivers positive outcomes and is more effective at forecasting rainfall.

Classifiers evaluations

Model	**Accuracy**	**Loss**
Architecture - 1	78%	0.4550
Architecture - 2	79%	0.4225
Architecture - 3	80%	0.4340
Architecture - 4	77.7%	0.4775
Architecture - 5	79.9%	0.4327

After experimenting the different architectures and comparing their accuracies, we decided to test the 3rd architecture using actual data, which helped us in seeing if the model actually works on predicting or not, since accuracy and loss aren't the only things, we should focus on when evaluating our model.

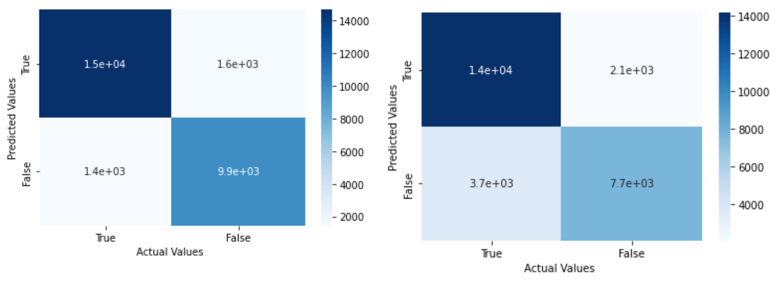


Fig.16. Confusion Matrix Comparisons between (a) Random Forest (b) ANN

The model was able to predict the true positives and true negative values precisely, by giving it a randomly chosen data on the same dataset, it was able to detect that the predicted value is zero or one by an 87% and a total of 80% of the data was detected correctly.
The proposed ANN model is evaluated with greater detail by making use of the classification report metrics, which can be seen in Figure.5. The resulting report of the ANN model is presented in table.2. It shows that the proposed model can correctly detect out of all the data a 87% chance that there won't be rain and a 68% chance that there will be rain. And out of all the days that actually did rain, the model was able to predict this outcome correctly by 80% for both zeros and ones, but the model is more beneficial in classifying no rain tomorrow.

D. *Interpretations and discussion*

Moreover, our proposed model was also tested using the Random Forest Machine Learning model after experimenting with the ANN issue. It achieved a 10% accuracy higher than our ANN model; it was also able to classify correctly if there is rain tomorrow or not. As seen in table.3, random forest model performs way better than the ANN and the reason to that is according to the findings, Neural Networks performs marginally better than Random Forests in case of performance. Any missing values can be handled correctly by the Random Forest model. Therefore, even with some missing input values, the Random Forest was still able to make correct predictions.

Deep learning can therefore result in significant advancements in extremely difficult fields. This includes speech recognition, image classification, etc. Random Forests have no chance of competing in these fields. But neural network models are better in terms of cost and time. [11]

TABLE I. ANN MODEL CLASSIFICATION REPORT

Class	**Classification Report**		
	Precision	***Recall***	***F1-Score***
0	0.79	0.87	0.83
1	0.79	0.68	0.73
Accuracy			0.79

TABLE II. RANDOM FOREST MODEL CLASSIFICATION REPORT

Class	**Classification Report**		
	Precision	***Recall***	***F1-Score***
0	0.91	0.90	0.91
1	0.86	0.87	0.87
Accuracy			0.89

V. RELATAD WORK

A. *Background*

Implementing Artificial Neural Networks (ANN), which are multi-layer fully connected neural nets that resemble the figure, was the main goal of our model approach. 15 below. They are made up of an output layer, several hidden layers, and an input layer that may contain numerous features. Each neuron in a layer is linked to every other neuron in the layer above it. [8] If you go deeper into the hidden/output neurons, you'll find that a non-linear activation function is applied after the weighted sum is multiplied by its inputs.

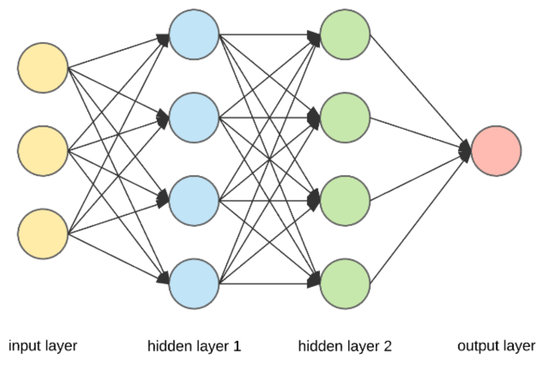


Fig.17. Artificial Neural Network Architecture

B. *Activation functions*

In order to determine the output of the neural network, such as yes or no, activation functions are used. The obtained values are mapped between 0 and 1 or -1 and 1. (depending upon the function). [8]
They can be categorized into two groups:

1- Initial Linear Activation Function
2- Non-Linear Activation Functions

One of the reasons why non-linear activation functions were developed was to assist us in generalizing or adapting the variety of data to differentiate between the output. The issue with linear is that it doesn't help with the complexity of various parameters of the data to be fed to the neural network. [9]
In our case, we have used the ReLu and Sigmoid nonlinear activation functions since our problem was focusing on classification and we had a huge number of input features.
The figure 16 shows a view of the different non-linear functions and their equations.

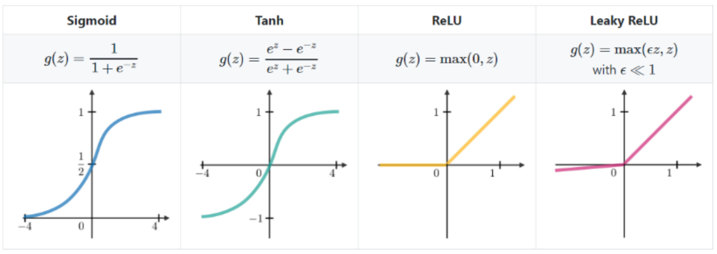


Fig.18. Non-Linear Activation Functions [10]

We used the sigmoid function in our output layer since it has a range of 0 to 1. Since the chance of anything exists only between the range of 0 and 1, sigmoid was the best option when trying to predict whether or not it would rain tomorrow on a given day with an output of either one or zero.
Relu, on the other hand, was utilized in the input and hidden layers and instantly turns any negative values into zero; this may occasionally compromise the model's capacity to accurately fit or train the data.
In conclusion, ANNs are very adaptable but also very strong deep learning models.

C. Comparisions

Our project relied on “Karnika Kapoor” work from Kaggle as
the theoretical framework. [7]
The difference between her analysis and ours is that during the data pre-processing part, we dealt with the imbalancement since it might affect the accuracy of the classification model but on the other hand, she skipped dealing with outliers, although it’s known that dealing with imbalancement in the attributes might affect the accuracy of the model.
Comparing the model creation process, we both chose to work with a sequential model API using Keras. The Sequential model allows us to build deep neural networks by stacking layers one on top of each other.

She had an input layer that consists of 32 different neurons due to her way of encoding the attributes which made half of the features duplicate, while we had only 20 neurons as input.
2 hidden layers were created on our model and one output layer, while she faced huge overfitting and had to regularize it using 2 dropout layers and early stopping but still didn’t make a big difference when plotting the accuracies against the epochs.
In Figure.11 you can check our model summary for clearer comparisons.

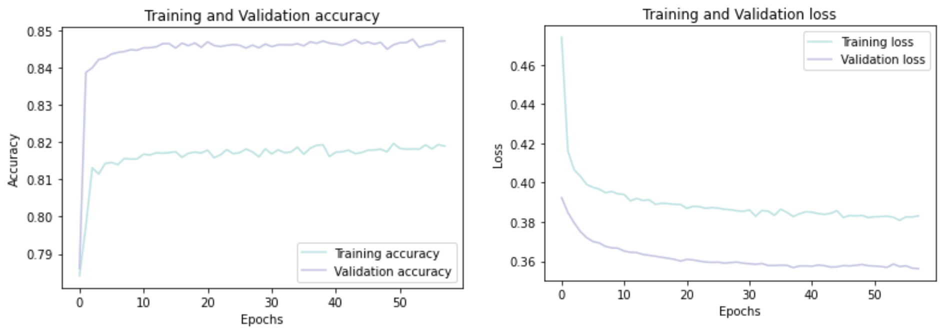


Fig.19. Karnika Kapoor’s Evaluations [7]

Comparing both classification reports, to measure the quality of predictions from a classification algorithm, it is one of the steps you can do afer your model as a part of evaluation. Although her model still got better results in getting True Positives and False Positives, our model is still in nearly the same range in classifying the outputs to its actual value. As seen in table.3. our model is still better in classifying availability of rain, unlike her model which got a 50% on recall.

Table III. Classification report Comparisons

Class	**Karnika Kapoor Classification Report**		
	Precision	***Recall***	***F1-Score***
0	0.88	0.94	0.91
1	0.71	0.50	0.58
Accuracy			0.85

Class	**Our Project Classification Report**		
	Precision	***Recall***	***F1-Score***
0	0.79	0.87	0.83
1	0.79	0.68	0.73
Accuracy			0.79

As seen above, although she had a higher accuracy of 82%, her model is still facing a huge noticeable overfitting. That makes our model being better in case of performance, we got 80% accuracy, but the overfitting is avoided.

VI. Conclusion

In this study, a new ANN architecture has been developed to forecast rainfall in Australia. After experimenting with various architectures and fine-tuning our model's hyperparameters, such as the number of neurons in the hidden layers, the number of epochs, and the batch size, we finally settled on a predictive algorithm that provided an accuracy score of 80%. The proposed model with the proposed hidden layer architecture and the processed dataset outperformed the three designs we implemented. By training the classification model on a time series problem, we can eventually increase the model's classification accuracy and forecast rainy days over the long run.

We have also obtained the knowledge of how to build, train and test deep learning models using Keras. How pre-processing the data can also be the problem with getting bad accuracy and inaccurate results. Selecting better features, getting cleaner data, and more samples are all important steps to do before taking a look at tuning the model for the aim of improving performance. The right amount of effort can get you the best results.

VII. References

[1] Len Calderone, “Weather Forecasting for the Farmer” Feb 2020, online: https://www.agritechtomorrow.com/article/2020/02/weather-forecasting-for-the-farmer/11981/#:~:text=By%20forecasting%20rainfall%20in%20a,the%20crop%20until%20rains%20come.

[2] Adamowski, J., Sun, K. “Development of a coupled wavelet transform and neural network method for flow forecasting of non-perennial rivers in semi-arid watersheds”,2010.

[3] Joe Young, “Rain in Australia”, Kaggle, 2010, online: https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package.

[4] Prena Nichani, “Appropriate way to treat Missing Values”, May 2, 2020, online: https://medium.com/analytics-vidhya/appropriate-ways-to-treat-missing-values-f82f00edd9be

[5] Shivam Kohli, “Understanding a classification report for your ML model”, Nov 18 2019, online: https://medium.com/@kohlishivam5522/understanding-a-classification-report-for-your-machine-learning-model-88815e2ce397

[6] Shrivastava, G., S. Karmakar, and M.K. Kowar, Application of Artificial Neural Networks in Weather Forecasting: A Comprehensive Literature Review. International Journal of Computer Applications, 2012.

[7] Karnika Kapoor, “Rain Prediction: ANN”, 2020, online: https://www.kaggle.com/code/karnikakapoor/rain-prediction-ann

[8] Arden Dertat, “Applied Deep Learning-Part1:ANN”, Aug 8, 2017, online: https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6

[9] Sagar Sharma, “Activation functions in neural networks”, Sep6,2017,online: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[10] Ladislas Nalborczyk, “ A gentle introduction to deep learning in R using Keras”, 21 May 2021, online: https://www.barelysignificant.com/slides/vendredi_quanti_2021/vendredi_quantis#1

[11] Prof. Dr. Peter Robach, “Neural Networks vs. Random Forests – Does it always have to be Deep Learning?”, Oct 2018, online:https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiRv8D9suL7AhW2aqQEHSwqBpEQFnoECA0QAw&url=https%3A%2F%2Fblog.frankfurt-school.de%2Fwp-content%2Fuploads%2F2018%2F10%2FNeural-Networks-vs-Random-Forests.pdf&usg=AOvVaw2kUyNHOz7DIBOWfvf8tucf