**6006CEM Machine Learning and Related Applications coursework. Australia Rainfall dataset**

**Gurvinder Nagra- SID 10248274**

**Coventry GitHub Repository URL** or **Coventry OneDrive URL** (mandatory):

<https://github.coventry.ac.uk/nagrag4/102482714-GN-s1>

**Table of contents**

Contents

[Introduction 2](#_Toc118378395)

[Analysing and pre-processing the data 2](#_Toc118378396)

[Applying different algorithms and methods to build learning models 6](#_Toc118378397)

[Making appropriate adjustments to improve the model’s performances 8](#_Toc118378398)

[Evaluating the models 8](#_Toc118378399)

[Comparing the approaches and results of other existing pieces of work on the same problem 13](#_Toc118378400)

[Bibliography 15](#_Toc118378401)

|  |
| --- |
| **Academic Report** |

# Introduction

The dataset we are analysing here is based upon Australia rainfall and whether it is likely that rain will fall, in this piece of work I will be conducting 2 machine learning algorithms upon this data set and checking the accuracy of whether it is likely to rain in Australia on the next day. I will be using 2 algorithmic approaches Logistic Regression and neural networks to do this. The reasoning behind this as stated, “Different algorithms like Neural Network, Sliding Window, Bayesian, Decision Tree, etc. have been used to develop such forecasting applications in today’s era” (LucilaOhno-Machado, 2002) And how “Linear and Logistic regression algorithms have been used to predict the weather on the basis of minimum and maximum temperatures, wind speed, humidity, precipitation and chances of rainfall.” (Sharma, 2019). By using these 2 different approaches I have the chance to analyse the data in 2 different ways giving me further insight to how these methods may differ, as both models work with the data differently and may provide different results as such.

There also have been similar approaches with different models such as Gaussian and Bernoulli models, though mine will require the data to be pre processed and worked with in a slightly different method to get our results.

# Analysing and pre-processing the data

When first starting with the data set, the first aim was to read it and display the data within it with basic built-in functions, to get a grasp of what was within the dataset and what type each variable in the data set was.

Graphical user interface, text, application

Description automatically generated

This is important since “real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.” (JavaTpoint, n.d.) When doing this I learned of all different data types and found the dataset was not clean and contained NaN values. I decided to start by sorting out the categorical variables and non-categorical variables into different lists to get an easier view of them.

Graphical user interface, text, application

Description automatically generated

Whilst doing that I also made sure to list out the number of unique values and null values so I knew how many would need to be sorted.

Graphical user interface, table

Description automatically generated

The next step was “process it and turn it into graphical representations that humans can use to quickly draw insights.” (Burns, 2020) which allowed me to better visualise the data, I did this with a bar chart and heatmap to find correlations within the entire dataset.

Chart, treemap chart

Description automatically generated

I had made sure to include more visualisations such as specific correlations which would be relevant after I cleaned up the data set. I did this firstly be making it so the variables RainTomorrow and RainToday would say 1 or 0 for whether it was a yes or no for rain meaning that it would be easier to read the data whilst coding it. Then for the missing values that I had discovered in the dataset I had decided to fill in the continuous variables with the mean anything missing and for object type variables I used mode since it was the only type that would work objects. After that I checked the values again and made sure there was no issues.

Text

Description automatically generated

After that I returned to the visualisation’s aspect , the first being comparing RainToday and RainTomorrow in a bar chart to see the data seems to be quite similar showing that the data seems to be able to predict rainfall somewhat accurately via comparison just from a simple view. I had also created some histograms to compare this in different locations which gave similar results. Once I had gathered this much, I moved onto sorting the dates within the data and comparing temperatures over the years with a data plot to see how temperatures made a difference and if other variables had a key play in rainfall.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface

Description automatically generated with low confidence

Encoding the data was the next step within the objects data types so that the machine learning algorithms we were about to use would be able to read them, as well as that I was sure to remove any outliers the graphs had shown to me.

Graphical user interface, text, application

Description automatically generated

# Applying different algorithms and methods to build learning models

When choosing the algorithms, I had decided to choose logistic regression as it is good for looking at historical data and predicting the likelihood of an event and since “Logistic regression is an example of supervised learning. It is used to calculate or predict the probability of a binary (yes/no) event occurring.” (Chandrasekaran, 2021). As well as that I had also decided upon neural networks as it is a more advanced algorithm why may delve into deeper results and the fact that “Compared to logistic regression, neural network models are more flexible, and thus more susceptible to overfitting. Network size can be restricted by decreasing the number of variables and hidden neurons, and by pruning the network after training.” (LucilaOhno-Machado, 2002).

I started by sorting the data I would be using for testing and created the variables x, which is the entire dataset, minus RainTomorrow, and the y which is RainTomorrow. After this I use test train split to train the data before putting it into the model, the reason for this being “ The idea of “sufficiently large” is specific to each predictive modeling problem. It means that there is enough data to split the dataset into train and test datasets and each of the train and test datasets are suitable representations of the problem domain. This requires that the original dataset is also a suitable representation of the problem domain.” (Brownlee, Train-Test Split for Evaluating Machine Learning Algorithms, 2020).

Firstly, I began with the Logistic Regression model inputting the variables that are needed and as well as that I set it up so it would be passed onto a confusion matrix for further analysis, I also was sure to get an F1 score to measure further accuracy onto the dataset. After that I moved onto plotting the confusion matrix as a heatmap to get a visualisation of it and then checking it in terms of true positive/negatives and false positives/negatives and check how many were predicted and how many were actual and with the heatmap I was able to see this better and understand what proved to be better. Next was the cross-validation score which would allow me to see if this model would be accurate and then averaged the scores that I had gotten to get a rough estimate on the final score for the logistic regression model.

Graphical user interface, text, application, email

Description automatically generated

Chart, treemap chart

Description automatically generated

Next the neural network, used mostly the same technique, having already trained the data I reused the training variables and inputted them into the model to get a testing and training score for it. With this I then inputted it into a confusion matrix printing out the classification report after putting into a heatmap for better visualisation. Whilst I also checked for true positives/negatives and false positives/negatives and plotting that to see my results. From there I got a cross validation score for this model as well after averaging all the different cross validation scores I got.

Graphical user interface, text, application

Description automatically generated

# Making appropriate adjustments to improve the model’s performances

After working with the models and analysing the results I had decided to make some improvements to the performance I did this by scaling the data this way I would be reshaping my data which would allow me to test bigger samples and see how accurate the results would be as such. I also did a rerun with tweaked parameters to see whether that would improve the performance which I did for the original run and the scaled run. As mentioned, before I also used cross validation whilst doing the models which will help prevent overfitting by testing other parts of the data by splitting it into groups, which will help evaluate the model’s effectiveness on unused and different parts of the data.

# Evaluating the models

From what can be seen of the logistic regression model that has been created the accuracy score is 84% showing that compared to the true labels the predicted labels within this model are close albeit with some errors showing the model has a decently good prediction rate.

To go further on this, I also calculated the f1 score of the model and plotting a confusion matrix to help visualise these results, this is because “F1-score is one of the most important evaluation metrics in machine learning. It elegantly sums up the predictive performance of a model by combining two otherwise competing metrics — precision and recall.” (LT, 2021), which takes the false positives/negatives and true positives/negatives into account, in this it was shown that the score is only 55% meaning that it wasn’t very accurate and predicting them. This would prove that the model’s binary classification systems are not as good as they could be and need improving to get a better result. However, I saw there were 20000 True positives however, in comparison to the other being within the 1000’s meaning that the data was able to correctly predict when it would rain meaning it was seemingly doing good. Which could argue that the f1 score may not have been the best indicator of performance here.

I also ran a cross validation score “to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.” (Brownlee, A Gentle Introduction to k-fold Cross-Validation, 2018). This averaged to be 84% as well, meaning that the model works well across different parts of the data that weren’t used during the model’s training showing the model has equal effectiveness regardless od the data and is able to generate decently accurate scores.

Chart, treemap chart

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Looking at the score for the neural network model we can see that it was 76% for both the training and testing data, we can see that this is a high score albeit not as good as logistic regression but still showing the neural network model is also able to get predictions with a good degree of precision. Just like for logistic regression I also got the true positives/negatives and false positives/negatives and in this case, it showed to have 16000 TP whilst having more true negatives and false positives adding to how this isn’t as accurate as logistic regression. Although generating close results shows that it has a good degrees of accuracy.

Graphical user interface, application, Teams

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

I also ran a cross validation score on the model which averaged out to 78.7% showing that it may has some variance with the data though from an average the mod will be the same accuracy with different data. However, looking at the individual scores it is shown that most of them are 84% whilst one is 55% which is bringing down the average, here we can evaluate the model may be having problems with skewed data and that the dataset may not be the best for this type of model. Overall, we can see that this model has a decent score at predicting rainfall though it does seem to have issues with certain parts of the data which would make us more inclined to use logistic regression over this for better results.

Graphical user interface, text, application, email

Description automatically generated

# Comparing the approaches and results of other existing pieces of work on the same problem

Looking at this dataset there are many different approaches that can be taken on this, some ranging from using different algorithms to get different results. One such algorithm is a decision tree classifier, it uses a similar approach with first reading the data, though there are different graphs plotted comparing more variables such as checking the temperature more precisely with bar charts and scatter plots as well as box plots, this allowed for a greater insight to how the data looks giving more insight on how to work with it. As well as that there is also more separation of the target variables using different methods albeit ending with the same result, I did but from there it starts by testing with Gaussian and Bernoulli models found on Kaggle (Marconato, 2022) using different data types they scaled. This led to giving scores of around 76-80% accuracy. Compared to the models I created, this shows that Gaussian model does outperform the Neural Networks and maybe considered a better solution to predicting RainTomorrow for this problem. Though for Bernoulli models it would seem just about equal to using neural networks for solving this problem. Overall, however we can see that neither of these approaches are able to beat logistic regression, though even by a small margin, it would seem that logistic regression and the pre processing methods that I used proved to yield better results.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

|  |
| --- |
| **Bibliography** |

# Bibliography

Brownlee, J. (2018, May 23). *A Gentle Introduction to k-fold Cross-Validation*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/k-fold-cross-validation/#:~:text=Cross%2Dvalidation%20is%20primarily%20used,the%20training%20of%20the%20model

Brownlee, J. (2020, July 24). *Train-Test Split for Evaluating Machine Learning Algorithms*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/

Burns, K. B. (2020, February ). *data visulization*. Retrieved from techtarget: https://www.techtarget.com/searchbusinessanalytics/definition/data-visualization#:~:text=Companies%20are%20increasingly%20using%20machine,in%20ways%20they%20can%20understand

Chandrasekaran, M. (2021, November 2021). *Logistic Regression for Machine Learning*. Retrieved from Capital One: https://www.capitalone.com/tech/machine-learning/what-is-logistic-regression/

JavaTpoint. (n.d.). *Data Processing in Machine Learning*. Retrieved from Javatpoint: https://www.javatpoint.com/data-preprocessing-machine-learning

LT, Z. (2021, Novemeber 23). *Essential Things You Need to Know About F1-Score*. Retrieved from Towards Data Science: https://towardsdatascience.com/essential-things-you-need-to-know-about-f1-score-dbd973bf1a3#:~:text=F1%2Dscore%20is%20one%20of,competing%20metrics%20%E2%80%94%20precision%20and%20recall

LucilaOhno-Machado, S. D. (2002, October). Logistic regression and artificial neural network classification models: a methodology review. In S. D. LucilaOhno-Machado, *Journal of Biomedical Informatics 35 (2002)* (pp. 352–359). ScienceDirect. Retrieved from ScienceDirect: https://www.sciencedirect.com/science/article/pii/S1532046403000340#:~:text=Compared%20to%20logistic%20regression%2C%20neural,pruning%20the%20network%20after%20training

Marconato, R. (2022, June). *Australia Weather EDA and ML*. Retrieved from Kaggle.com: https://www.kaggle.com/code/raphaelmarconato/australia-weather-eda-and-ml

Sharma, A. (2019, December). Retrieved from IOE Graduate Conference: http://conference.ioe.edu.np/publications/ioegc2019-winter/IOEGC-2019-Winter-33.pdf

|  |
| --- |
| **Appendix B** |

**import** **sklearn.metrics**

**from** **sklearn.metrics** **import** precision\_score, recall\_score, confusion\_matrix, classification\_report, accuracy\_score, f1\_score

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

**import** **os**

**import** **missingno** **as** **msno**

**import** **seaborn** **as** **sns**

**import** **matplotlib.pyplot** **as** **plt**

**from** **sklearn** **import** preprocessing

**from** **sklearn.model\_selection** **import** train\_test\_split

**from** **scipy** **import** stats

**from** **sklearn.linear\_model** **import** LogisticRegression

**from** **sklearn.linear\_model** **import** LinearRegression

**from** **imblearn.over\_sampling** **import** SMOTE

**from** **collections** **import** Counter

**from** **sklearn.metrics** **import** confusion\_matrix

**from** **sklearn.metrics** **import** accuracy\_score, f1\_score

**import** **plotly.express** **as** **px**

**from** **sklearn.svm** **import** SVC

**import** **warnings**

warnings.filterwarnings("ignore")

df = pd.read\_csv('weatherAUS.csv')

df.head()

df.info()

Categorical\_Variables, Contin\_Val=[],[]

**for** i **in** df.columns:

**if** df[i].dtype == 'object':

Categorical\_Variables.append(i)

**else**:

Contin\_Val.append(i)

**print**(Categorical\_Variables)

**print**(Contin\_Val)

df.nunique()

df.isnull().sum()

msno.bar(df, sort='ascending')

plt.figure(figsize=(17,15))

ax = sns.heatmap(df.corr(), square=True, annot=True, fmt='.2f')

ax.set\_xticklabels(ax.get\_xticklabels(), rotation=90)

plt.show()

df['RainTomorrow'] = df['RainTomorrow'].map({'Yes': 1, 'No': 0})

df['RainToday'] = df['RainToday'].map({'Yes': 1, 'No': 0})

(df.isnull().sum()/len(df))\*100

*#Filling the missing values for continuous variables with mean*

df['MinTemp']=df['MinTemp'].fillna(df['MinTemp'].mean())

df['MaxTemp']=df['MaxTemp'].fillna(df['MaxTemp'].mean())

df['Rainfall']=df['Rainfall'].fillna(df['Rainfall'].mean())

df['Evaporation']=df['Evaporation'].fillna(df['Evaporation'].mean())

df['Sunshine']=df['Sunshine'].fillna(df['Sunshine'].mean())

df['WindGustSpeed']=df['WindGustSpeed'].fillna(df['WindGustSpeed'].mean())

df['WindSpeed9am']=df['WindSpeed9am'].fillna(df['WindSpeed9am'].mean())

df['WindSpeed3pm']=df['WindSpeed3pm'].fillna(df['WindSpeed3pm'].mean())

df['Humidity9am']=df['Humidity9am'].fillna(df['Humidity9am'].mean())

df['Humidity3pm']=df['Humidity3pm'].fillna(df['Humidity3pm'].mean())

df['Pressure9am']=df['Pressure9am'].fillna(df['Pressure9am'].mean())

df['Pressure3pm']=df['Pressure3pm'].fillna(df['Pressure3pm'].mean())

df['Cloud9am']=df['Cloud9am'].fillna(df['Cloud9am'].mean())

df['Cloud3pm']=df['Cloud3pm'].fillna(df['Cloud3pm'].mean())

df['Temp9am']=df['Temp9am'].fillna(df['Temp9am'].mean())

df['Temp3pm']=df['Temp3pm'].fillna(df['Temp3pm'].mean())

*#These can only be filled with mode due to datatype*

df['RainToday']=df['RainToday'].fillna(df['RainToday'].mode()[0])

df['RainTomorrow']=df['RainTomorrow'].fillna(df['RainTomorrow'].mode()[0])

df['WindDir9am'] = df['WindDir9am'].fillna(df['WindDir9am'].mode()[0])

df['WindGustDir'] = df['WindGustDir'].fillna(df['WindGustDir'].mode()[0])

df['WindDir3pm'] = df['WindDir3pm'].fillna(df['WindDir3pm'].mode()[0])

(df.isnull().sum()/len(df))\*100

*#Visulisation of rain today and tomorrow*

fig, ax =plt.subplots(1,2)

**print**(df.RainToday.value\_counts())

**print**(df.RainTomorrow.value\_counts())

plt.figure(figsize=(20,20))

sns.countplot(data=df,x='RainToday',ax=ax[0])

sns.countplot(data=df,x='RainTomorrow',ax=ax[1])

fig = px.histogram(df, x="Rainfall", title="Rainfall in mm", color\_discrete\_sequence=["blue"], nbins=25)

fig.show()

fig = px.histogram(df, x='Location', title='RainToday In Different Locations', color='RainToday')

fig.show()

fig = px.histogram(df, x='Location', title='RainTomorrow In Different Locations', color='RainTomorrow')

fig.show()

df['Date'] = pd.to\_datetime(df['Date'])

df['Year'] = df['Date'].dt.year

df['Year'].head()

df['Month'] = df['Date'].dt.month

df['Month'].head()

df['Day'] = df['Date'].dt.day

df['Day'].head()

df\_dateplot = df.iloc[-900:,:]

plt.figure(figsize=[20,5])

plt.plot(df\_dateplot['Date'],df\_dateplot['MinTemp'],color='blue',linewidth=1, label= 'MinTemp')

plt.plot(df\_dateplot['Date'],df\_dateplot['MaxTemp'],color='red',linewidth=1, label= 'MaxTemp')

plt.fill\_between(df\_dateplot['Date'],df\_dateplot['MinTemp'],df\_dateplot['MaxTemp'], facecolor = '#EBF78F')

plt.title('MinTemp vs MaxTemp by Date')

plt.legend(loc='lower left', frameon=False)

plt.show()

df.drop('Date', axis=1, inplace = True)

le = preprocessing.LabelEncoder()

df['Location'] = le.fit\_transform(df['Location'])

df['WindDir9am'] = le.fit\_transform(df['WindDir9am'])

df['WindDir3pm'] = le.fit\_transform(df['WindDir3pm'])

df['WindGustDir'] = le.fit\_transform(df['WindGustDir'])

*#removing outliers*

df=df[(np.abs(stats.zscore(df)) < 3).all(axis=1)]

df.shape

X = df.drop(["RainTomorrow"], axis=1)

y = df["RainTomorrow"]

*# Splitting test and training sets*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 42)

model = LogisticRegression(max\_iter=500)

model.fit(X\_train, y\_train)

predicted=model.predict(X\_test)

conf = confusion\_matrix(y\_test, predicted)

**print** ("The accuracy of Logistic Regression is : ", accuracy\_score(y\_test, predicted))

**print**()

**print**("F1 score for logistic regression is :",f1\_score(y\_test, predicted,))

y\_pred = model.predict(X\_test)

cmap1 = sns.diverging\_palette(260,-10,s=50, l=75, n=5, as\_cmap=True)

plt.subplots(figsize=(12,8))

cf\_matrix = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(cf\_matrix/np.sum(cf\_matrix), cmap = cmap1, annot = True, annot\_kws = {'size':15})

**print**(classification\_report(y\_test, y\_pred))

cm = confusion\_matrix(y\_test, y\_pred)

**print**('Confusion matrix**\n\n**', cm)

**print**('**\n**True Positives(TP) = ', cm[0,0])

**print**('**\n**True Negatives(TN) = ', cm[1,1])

**print**('**\n**False Positives(FP) = ', cm[0,1])

**print**('**\n**False Negatives(FN) = ', cm[1,0])

cm\_matrix = pd.DataFrame(data = cm, columns = ['Actual Positive:1', 'Actual Negative:0'],

index=['Predict Positive:1', 'Predict Negative:0'])

cm\_matrix.head()

sns.heatmap(cm\_matrix, annot=True, fmt='d', cmap='YlGnBu');

**from** **sklearn.model\_selection** **import** cross\_val\_score

scores = cross\_val\_score(model, X\_train, y\_train, cv = 5, scoring='accuracy')

**print**('Cross-validation scores:{}'.format(scores))

**print**('Average cross-validation score: {:.4f}'.format(scores.mean()))

**from** **sklearn.neural\_network** **import** MLPClassifier

model = MLPClassifier(random\_state=0)

model.fit(X\_train, y\_train)

score= model.score(X\_test, y\_test)

**print**('Testing score: {:.3f}'.format(score))

score = model.score(X\_train, y\_train)

**print**("Training score: {:.3f}".format(score))

y\_pred = model.predict(X\_test)

cmap1 = sns.diverging\_palette(260,-10,s=50, l=75, n=5, as\_cmap=True)

plt.subplots(figsize=(12,8))

cf\_matrix = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(cf\_matrix/np.sum(cf\_matrix), cmap = cmap1, annot = True, annot\_kws = {'size':15})

**print**(classification\_report(y\_test, y\_pred))

cm = confusion\_matrix(y\_test, y\_pred)

**print**('Confusion matrix**\n\n**', cm)

**print**('**\n**True Positives(TP) = ', cm[0,0])

**print**('**\n**True Negatives(TN) = ', cm[1,1])

**print**('**\n**False Positives(FP) = ', cm[0,1])

**print**('**\n**False Negatives(FN) = ', cm[1,0])

cm\_matrix = pd.DataFrame(data = cm, columns = ['Actual Positive:1', 'Actual Negative:0'],

index=['Predict Positive:1', 'Predict Negative:0'])

cm\_matrix.head()

sns.heatmap(cm\_matrix, annot=True, fmt='d', cmap='YlGnBu');

**from** **sklearn.model\_selection** **import** cross\_val\_score

scores = cross\_val\_score(model, X\_train, y\_train, cv = 5, scoring='accuracy')

**print**('Cross-validation scores:{}'.format(scores))

**print**('Average cross-validation score: {:.4f}'.format(scores.mean()))