Machine learning

Final project

Omer Ofir 203769187

Roee Zvi 316491661

**Task:**

חיזוי גשם למחרת על ידי אימון מודלים של סיווג על משתנה היעד

תוֹכֶן:

מערך נתונים זה מכיל כ-10 שנים של תצפיות מזג אוויר יומיות ממקומות רבים ברחבי אוסטרליה.

הוא משתנה היעד לניבוי RainTomorrow

. זה אומר -- האם ירד גשם למחרת, כן או לא? עמודה זו היא כן אם הגשם באותו יום היה 1 מ"מ או יותר.

עליך בנות אלגוריתם חיזוי לפי הנתונים הצורפים

**Description:**

The task is to define if it will rain tomorrow in Australia. for this task, we are using the NN TensorFlow model and SKleran tools. This is a neural network that uses binary classification to predict whether, given meteorological observations of a given day at a given weather station in Australia, it will rain there the next day. The model is trained and tested on a dataset containing about 10 years of daily weather observations from numerous Australian weather stations.

As we said, we decided to implement this project with Tensorflow 2 and Keras and SKlearn.

**Milestones**

* **Data Pre-processing -** get ready the data set.
* **Build the TensorFlow model -** define the layers, training set/learning set, Beach size and test data size.
* **Training the model -** define the number of Epoch.
* **Predict** - input the same data - predict if it will rain tomorrow.

**Data Pre-processing**

In this step, we organize the data, delete the unnecessary column, and create new data set to work with.

A picture containing text

Description automatically generatedFor this task, we build the python code: preprocessing.py

The first thing that we need to do is to load the data file, then, we create the new data set (.csv).

Text

Description automatically generatedDefine the input column and the columns we want to use, then, define all the numeric labels and the non-numeric label.

The next step is to remove all the rows that contains NaN (Reduce from 142193 rows to 56420 rows.).

Convert all the 'Yes/No' labels to 1/0, Scale/normalize numeric columns by calculating the z-score of each value.

Graphical user interface, table

Description automatically generatedAt the end of the process, the code will create new .csv file called 'data.csv

**Build the TensorFlow model**

The first thing we need to do is to call all the libraries:

Text

Description automatically generated

As you can see, we have used the TensorFlow and Keras libraries and a couple more libraries for execution.

Text

Description automatically generated

The number lines that we take from the file.

The number of batch (how many lines the ML learn in peralla).

Text

Description automatically generated

Define the batch size and the data set size (we don't need to use all the data sets).

build the NN and use "Adam" binary decision.

Diagram, schematic

Description automatically generated

**Training the model**

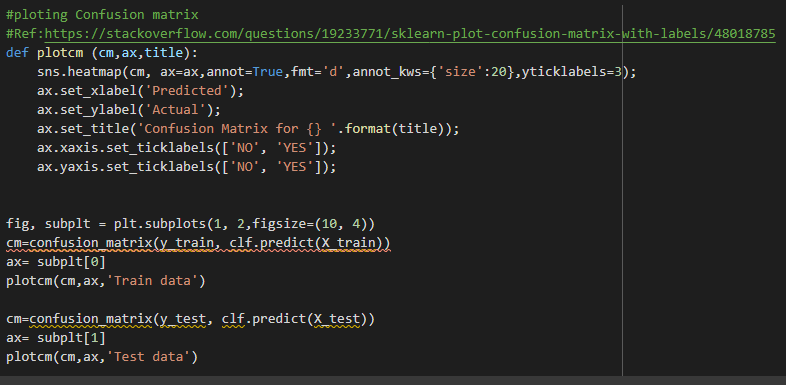
Text

Description automatically generated

Text

Description automatically generatedFor this run use only 6 epochs. We have called the .fit function (from Kares) to start the training).

As you can see, we got an accuracy of 87% and a loss of 30% (if we will train the algorithm more we can get better results).



Chart, waterfall chart

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Predict-**

for the prediction we have used 2 data, 1 from the train and one from the test. for the training dataset, the algorithm needs to predict 100 % because the algorithm knows the data, the test dataset is the real test.

Text

Description automatically generated

Text

Description automatically generated

We have used 1 batch (that includes 5 rows from the dataset). and the predicted results are at the bottom.

For the test set:

Text

Description automatically generated Text

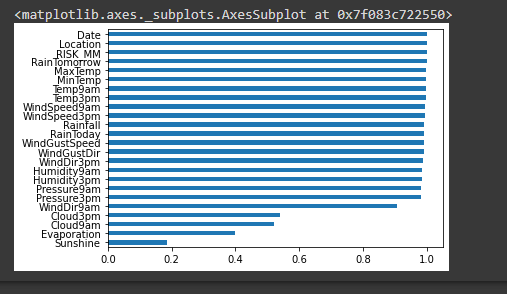
Description automatically generated

The full code is an attached.

**Second method -Sklearn**

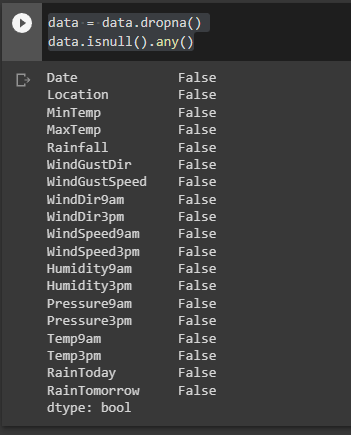
The type of machine learning we will be doing is called classification, because when we make predictions, we are classifying each day as rainy or not. More specifically, we are performing binary classification, which means that there are only two different states we are classifying.

Null values Let's get rid of columns with significant amount of null values. And in the rest columns we will drop rows with null values.



Cloud9pm, Cloud3pm, Evaporation, and Sunshine must be dropped since significant amount of records in these columns is missed. Also, we should exclude RISK\_MM because it can leak the answers to the model and reduce its predictability.

Let's drop rows with null values in them.

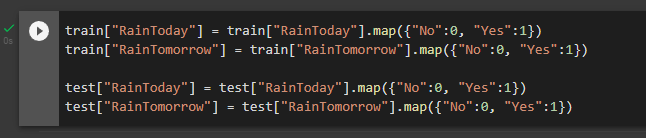


Text

Description automatically generatedSplit into train and test, we must be aware of one important thing: any change we make to the train data, we also need to make to the test data, otherwise we will be unable to use our model.

Deal with categorical variables, to apply such algorithms as Logistic Regression we need to convert the non-numeric data into numeric data. Categorical variables with only 2 possible values can be converted into variables with 0s and 1s as values. For categorical variables with 3 and more possible value we will create dummy variables.

Convert values in columns "RainToday" and "RainTomorrow" from "No" and "Yes" to 0 and 1



Visualization of how categorical variables impact on forming tomorrow's rain.

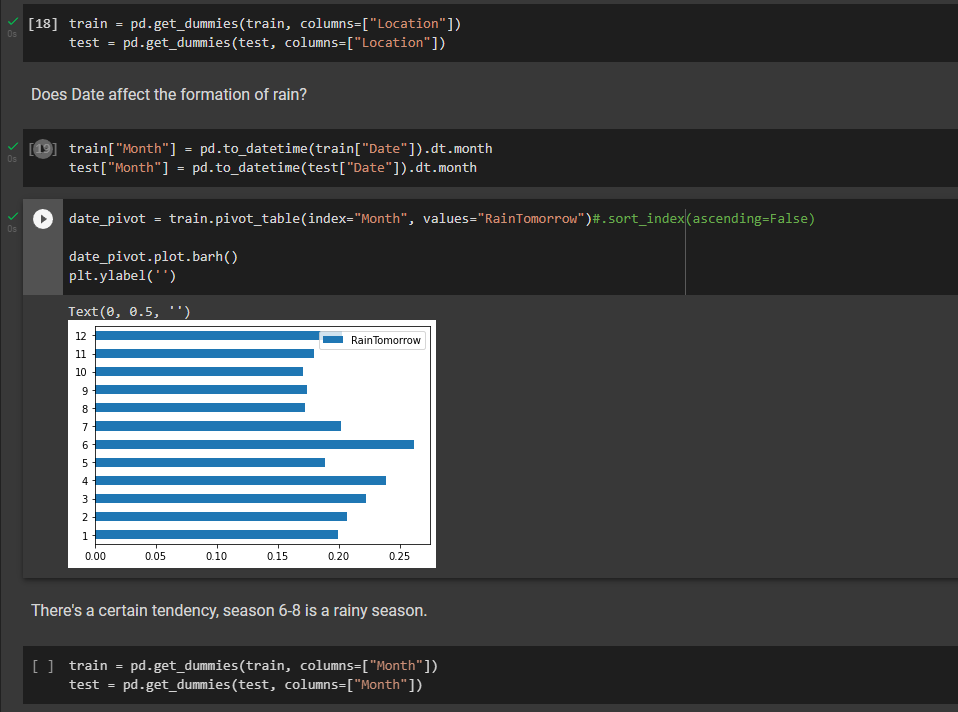
Chart, bar chart

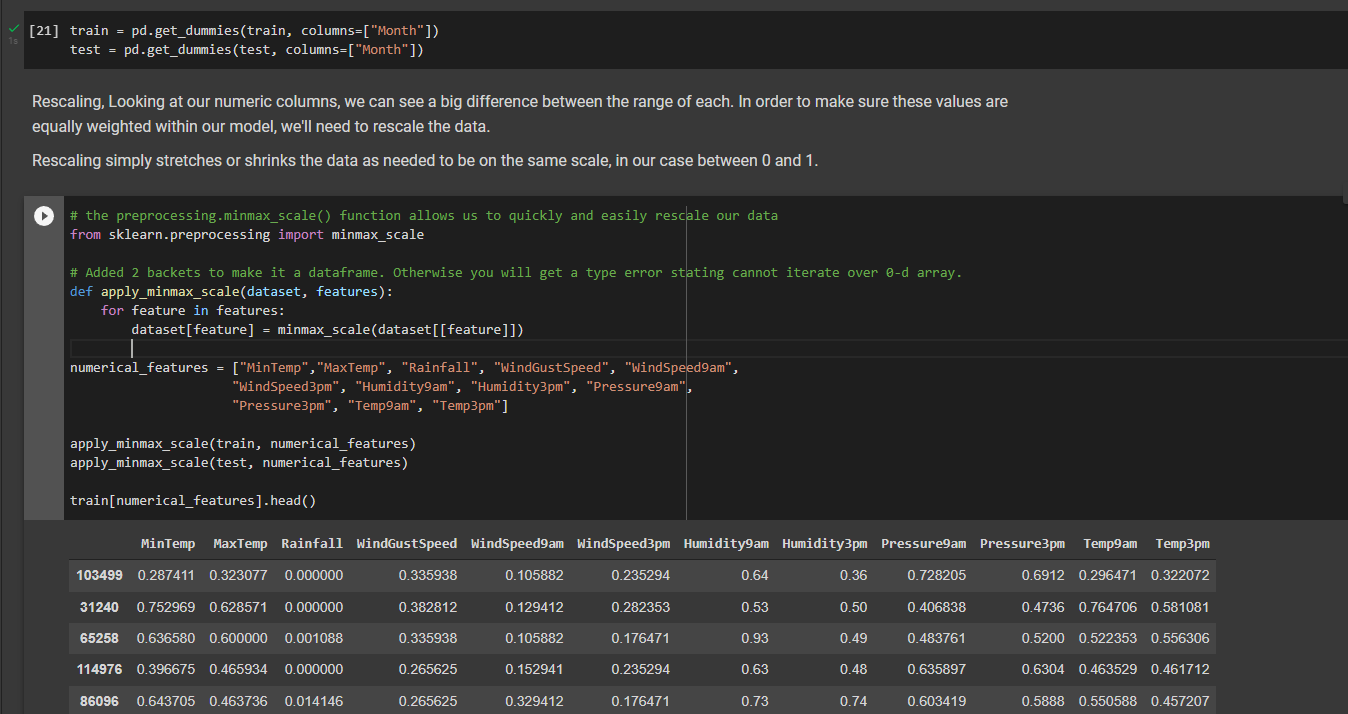
Description automatically generated

Chart, bar chart

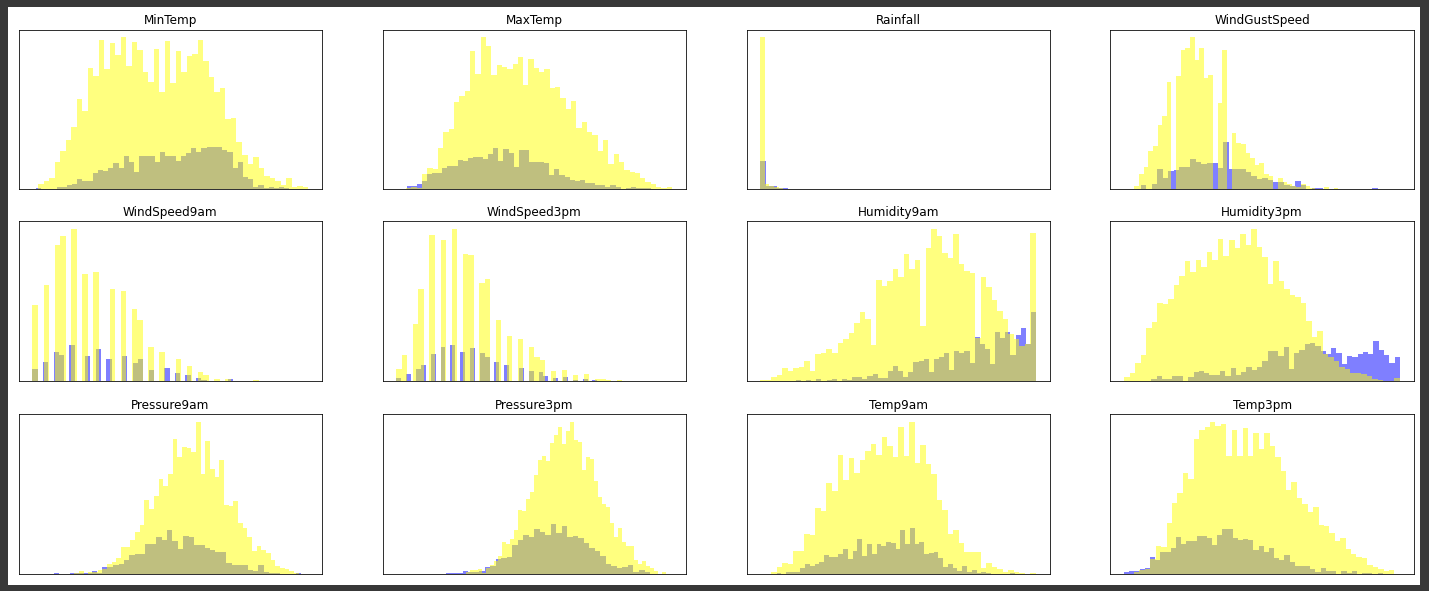
Description automatically generated

Yes, Location obviously affects the formation of tomorrow's rain! So, we're going to use this variable, and to use this categorical variable we must create dummies.



There's a certain tendency, season 6-8 is a rainy season.

Visualization of how numerical variables impact on forming tomorrow's rain.

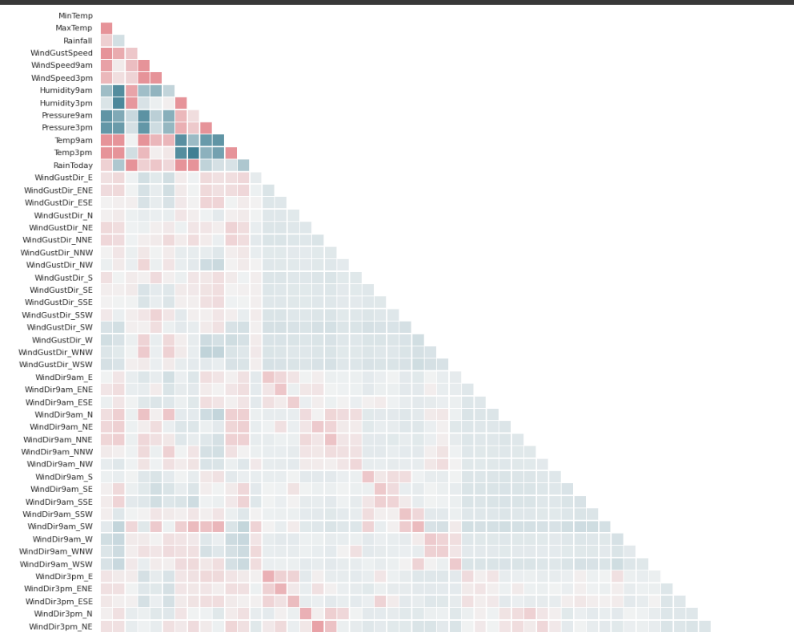


We are interested in variables with plots where blue and yellow areas have different shapes. Such variables have impact (positive or negative) on forming tomorrow's rain. The most obvious one is Humidity3pm! The rest is not that clear, we will use another feature selection method.

Collinearity, we now have 73 possible feature columns we can use to train our model. One thing to be aware of as you start to add more features is a concept called collinearity. Collinearity occurs where more than one feature contains data that are similar.

The effect of collinearity is that your model will overfit - you may get great results on your test data set, but then the model performs worse on unseen data (like the test set).

A common way to spot collinearity is to plot correlations between each pair of variables in a heatmap.



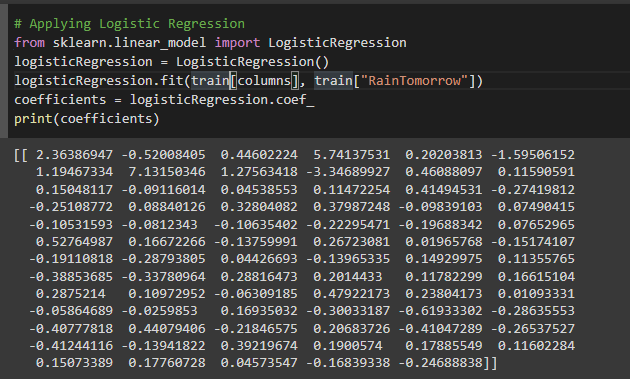
We can see that there is correlation about 30-50% between some variables. That's not enough to remove one of them and rely on the other.

Apart from that, we should remove one of each of our dummy variables to reduce the collinearity in each. We'll remove:

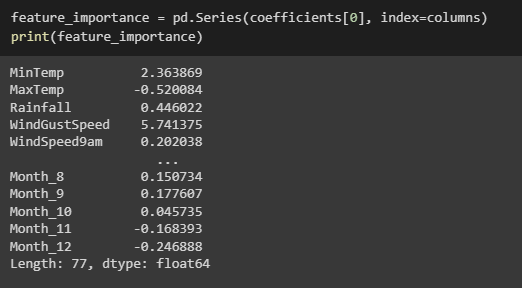
WindGustDir\_E WindDir9am\_E WindDir3pm\_E

Feature selection to select the best-performing features, we need a way to measure which of our features are relevant to our outcome - in this case, the impact on forming tomorrow's rain. One effective way is by training a logistic regression model using all our features, and then looking at the coefficients of each feature.

The scikit-learn Logistic Regression class has an attribute in which coefficients are stored after the model is fit, LogisticRegression.coef\_. We first need to train our model, after which we can access this attribute.



The coef() method returns a NumPy array of coefficients, in the same order as the features that were used to fit the model. To make these easier to interpret, we can convert the coefficients to a pandas series, adding the column names as the index:

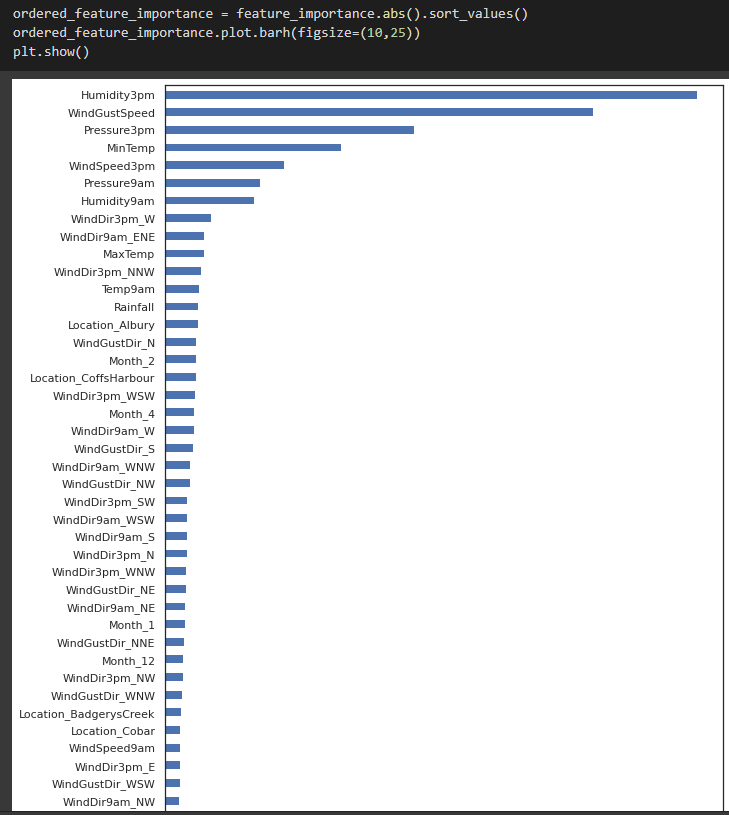


A picture containing table

Description automatically generated

The plot we generated shows a range of both positive and negative values. Whether the value is positive or negative isn't as important in this case, relative to the magnitude of the value. If you think about it, this makes sense. A feature that indicates strongly whether it’s not going to rain tomorrow is just as useful as a feature that indicates strongly that it’s going to rain tomorrow, given they are mutually exclusive outcomes.

To make things easier to interpret, we'll alter the plot to show all positive values, and have sorted the bars in order of size:



We'll train a model with the top 4 scores.

