Will it Rain Tomorrow in Australia?

Saheed A.Tijani; Feb. 2019

The full Jupyter note for this project is available **here**

Objective:

To predict whether or not it will rain anywhere in Australia tomorrow.

Why do this?

Below are some use cases of this information:

- Planning of irrigation and other farming activities
- Airplane route planning
- Construction activities scheduling
- Extracurricular activities planning for school children

About the Dataset:

- Data was captured daily from numerous weather stations across Australia between 2007 and 2017.
- Contains 23 independent variables and 142,193 observations.
- There was a sizeable class imbalance, this prompted me to use the AUC ROC metric in model evaluation.

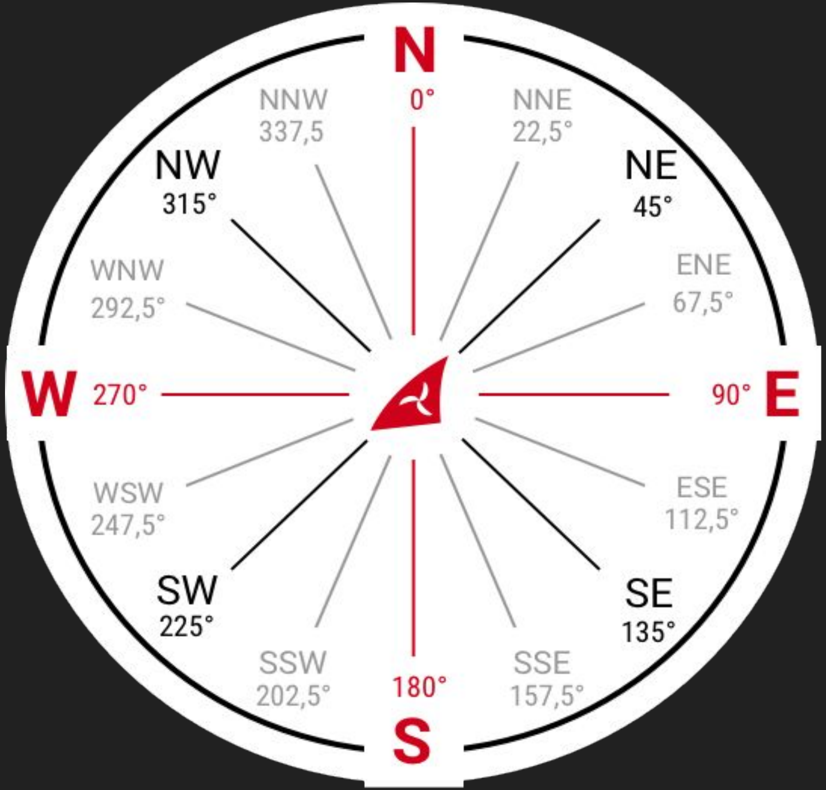
See more details about data and it’s source here

Features in dataset and their types

FEATURE NAME	TYPE
Date	Date
Location	String/Categorical
MinTemp	Numerical
MaxTemp	Numerical
Rainfall	Numerical
Evaporation	Numerical
Sunshine	Numerical
WindGustDir	String/Categorical
WindGustSpeed	String/Categorical
WindDir9am	String/Categorical
WindDir3pm	String/Categorical
WindSpeed9am	Numerical
WindSpeed3pm	Numerical
Humidity9am	Numerical
Temp9am	Numerical
Temp3pm	Numerical
RainToday	Boolean

RISK_MM	Numerical
RainTomorrow	Boolean
Humidity3pm	Numerical
Pressure9am	Numerical
Pressure3pm	Numerical
Cloud9am	Numerical
Cloud3am	Numerical

Feature Engineering

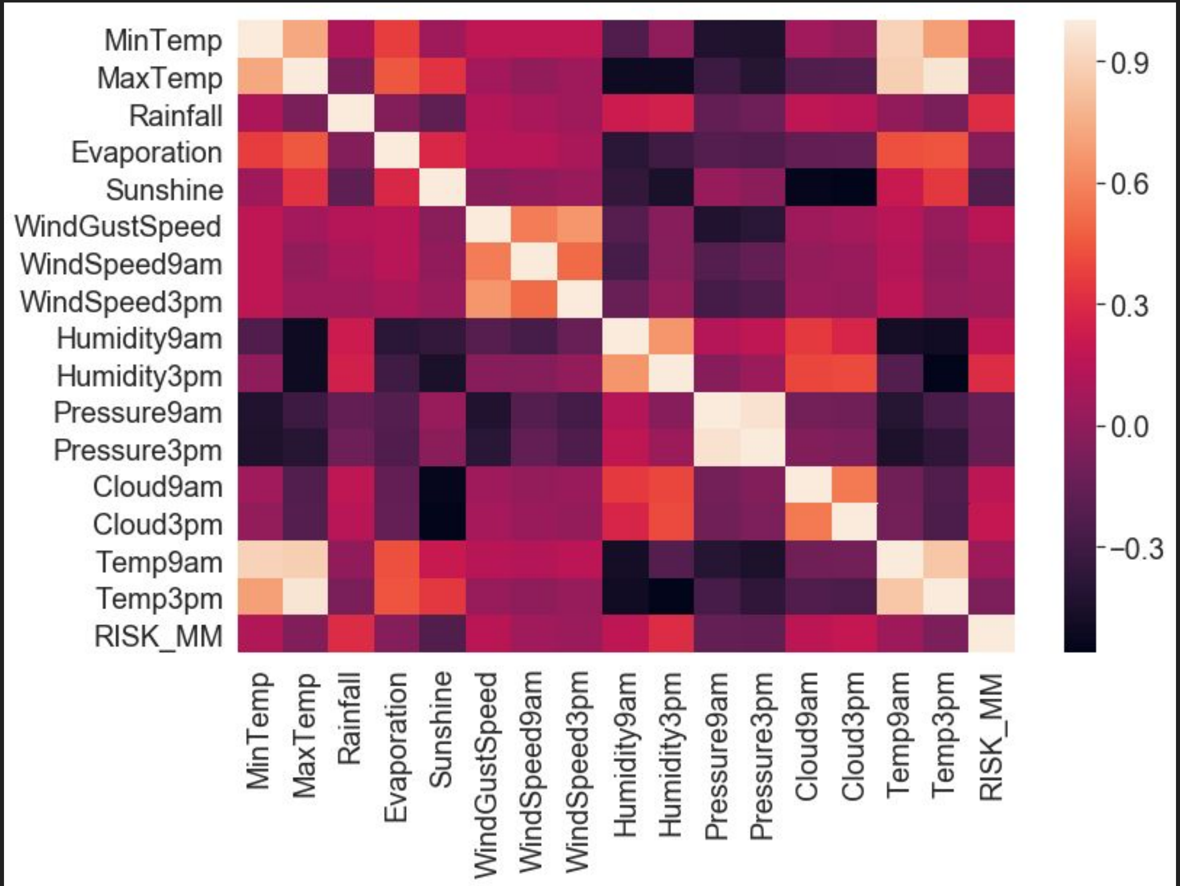


Standard clockwise cardinal rotation 0 - 360

- I ranked the values for the wind direction features from 0 - 15 using standard clockwise cardinal rotation.

- I one-hot-encoded location for the first model iteration for the sake of simplicity.

Correlation Assessment of Numerical Features



<u>High Correlation</u>

- Humidity vs Rainfall
- Temperature vs Evaporation
- Humidity vs Cloud

Benchmark Model: Decision Tree(DT)

MODEL REPORT:

Train	Test
Accuracy: 1.0	Accuracy:0.7959
AUC score:1.0	AUC score:0.7056

CV_scores (metric: AUC)
Mean: 0.7050
Std: 0.0042

As expected with a typical DT model, it recorded a 100% training accuracy. But model seems to generalise well and stable considering the CV scores.

Confusion Matrix - Train

98710	0
0	27998

Confusion Matrix - Test

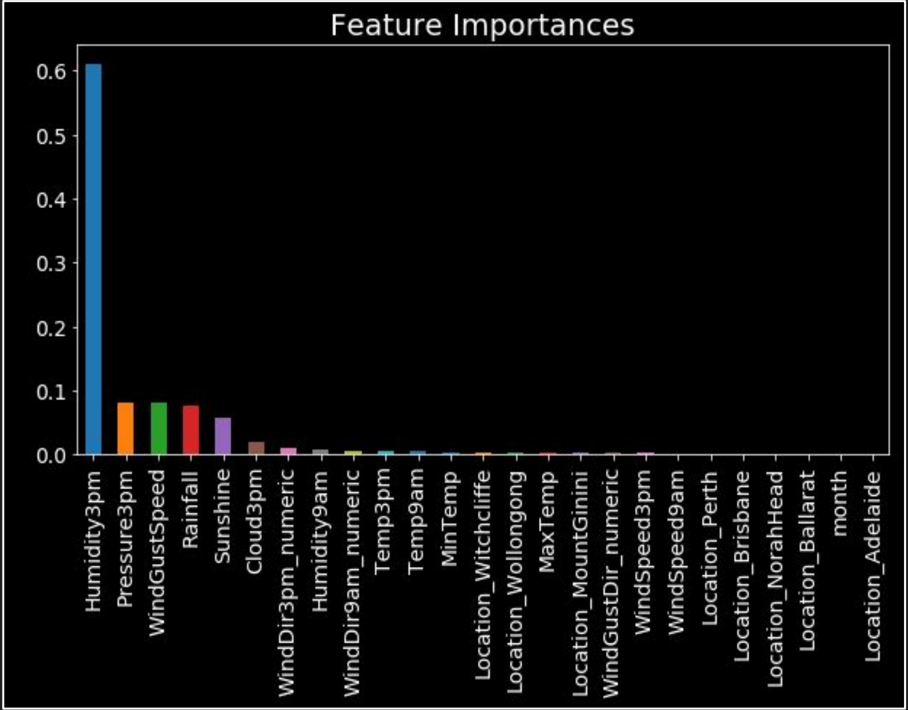
9348	1528
1468	1735

Default Gradient Boost Model

```
MODEL REPORT – Default BGM:

Train              Test
Accuracy:0.8541    Accuracy:0.8545
AUC score: 0.8803  AUC score:0.8788

Cross validation scores:
Mean - 0.8768
Std - 0.0023
```



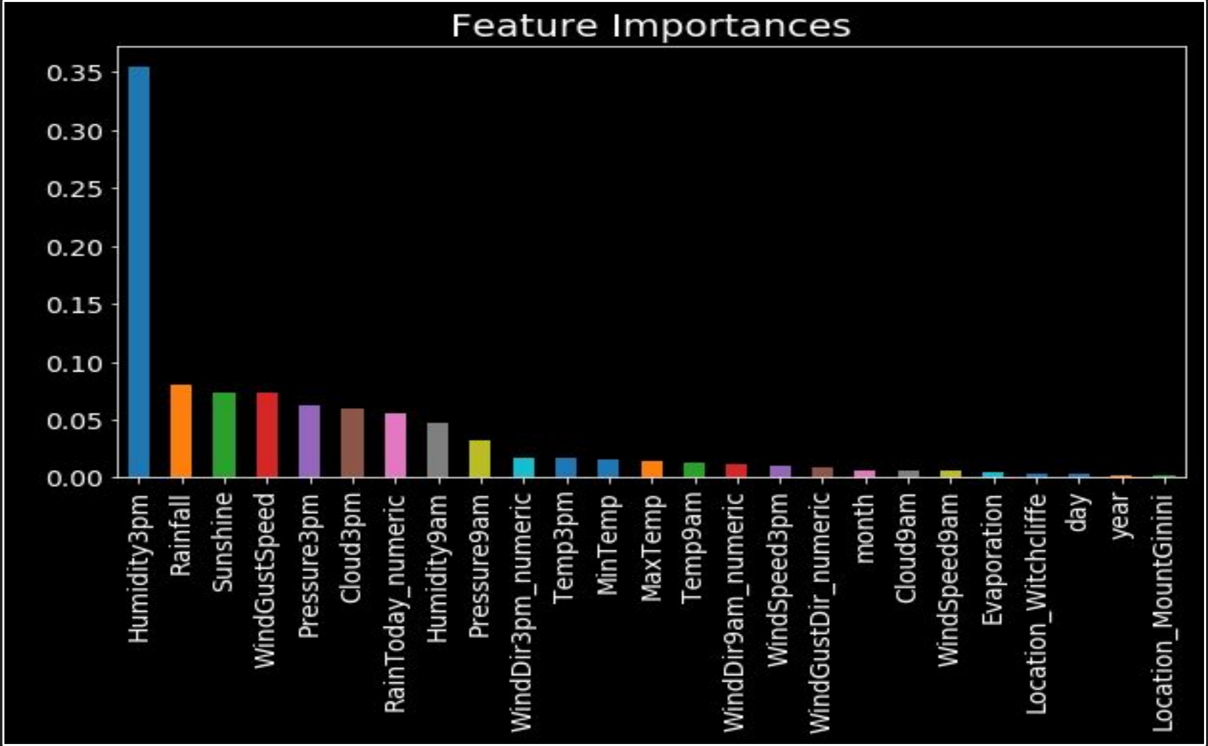
- Compared to the DT, there's a significant performance improvement here.
- On the flip side, about 60% of the variance in the outcome was derived from only **Humidity3pm**. This was regularized in the tuned model.

Tuned Gradient Boost Model:

Tuned param values

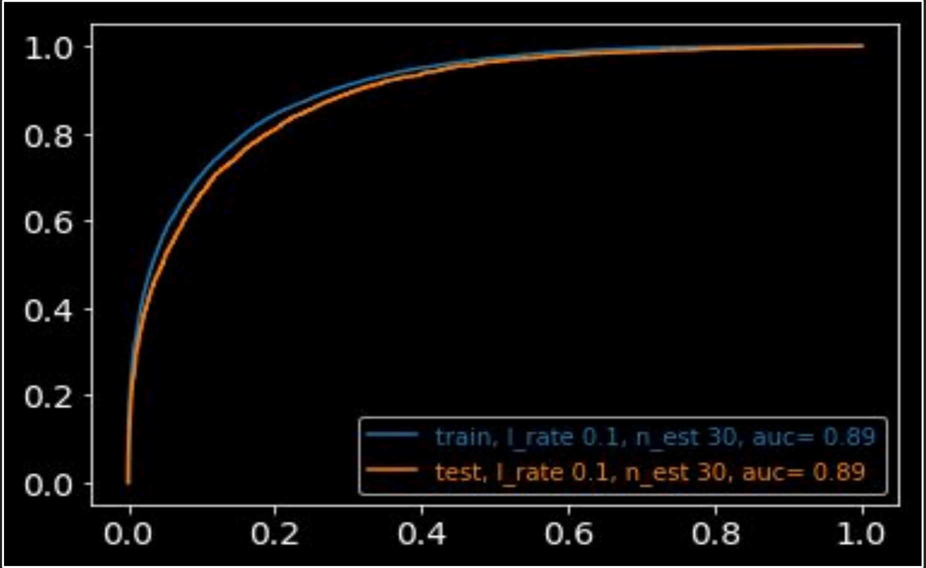
n_estimators	30
max_depth	15
min_samples_split	600
min_samples_leaf	100
subsample	0.7
max_features	19
random_state	10

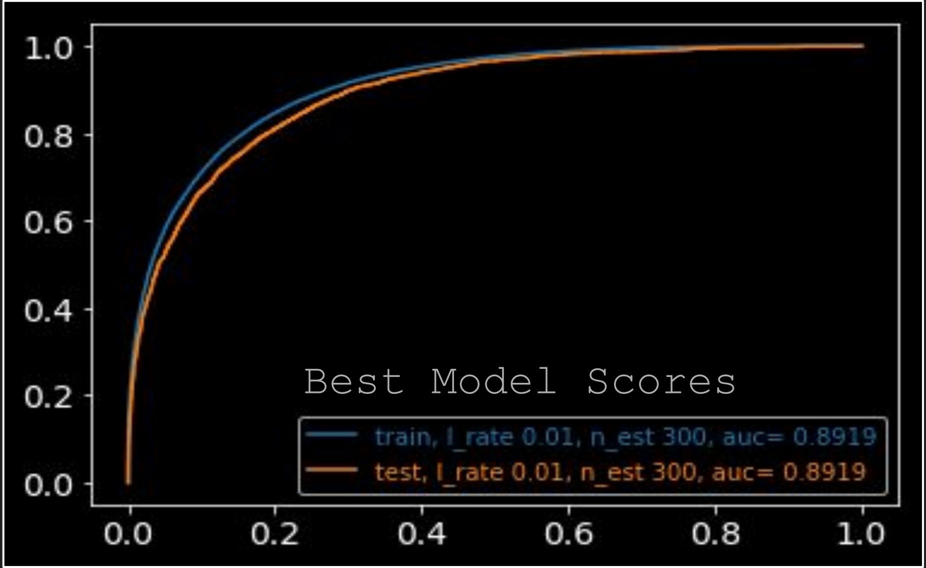
More distributed feature importances



Tuned Gradient Boost Model Scores

Learning_rate(l_rate)	n_est	accuracy	cv_mean	cv_std	AUC_score
0.10	30	0.8645	0.8859	0.0019	0.9063
0.01	300	0.8657	0.8880	0.0020	0.9088





Model Comparison - Logistic Regression

Running the data through Logistic Regression, the model performed worse than BGM. The accuracy scores peaked at 0.8499 with c=10.

Reg. parameter	accuracy	cv_maean	cv_std	AUC_score
0.001	0.8220	0.8370	0.0014	0.8406
0.010	0.8435	0.8603	0.0013	0.8618
0.100	0.8485	0.8682	0.0013	0.8492
1.000	0.8497	0.8712	0.0012	0.8718
10.000	0.8499	0.8716	0.0012	0.8721
100.000	0.8499	0.8716	0.001	0.8721

Areas of Future Improvement

- Further work could be done around feature engineering. For example, location could be converted into coordinates(i.e latitude and longitude) of weather stations rather than encoding as dummies.

- Introducing elements of time and trends such as seasonality. For example we could dummify the time of day and the season of the year from when an observation was recorded.

- I could try more advanced models like Neural Networks.

Thank you for your time….

Appendix

Navigating the Jupyter Notebook:

In [1 - 3] – Tools and Dataset importation

In [4 – 17] – Initial Data Exploration and Fixing Nans

In [18 – 32] – Feature Engineering and Further Data Exploration

In [19 – 34] – Data Preparation and Ground Work for Modelling

In [62 – 65] – Baseline Models and Evaluation

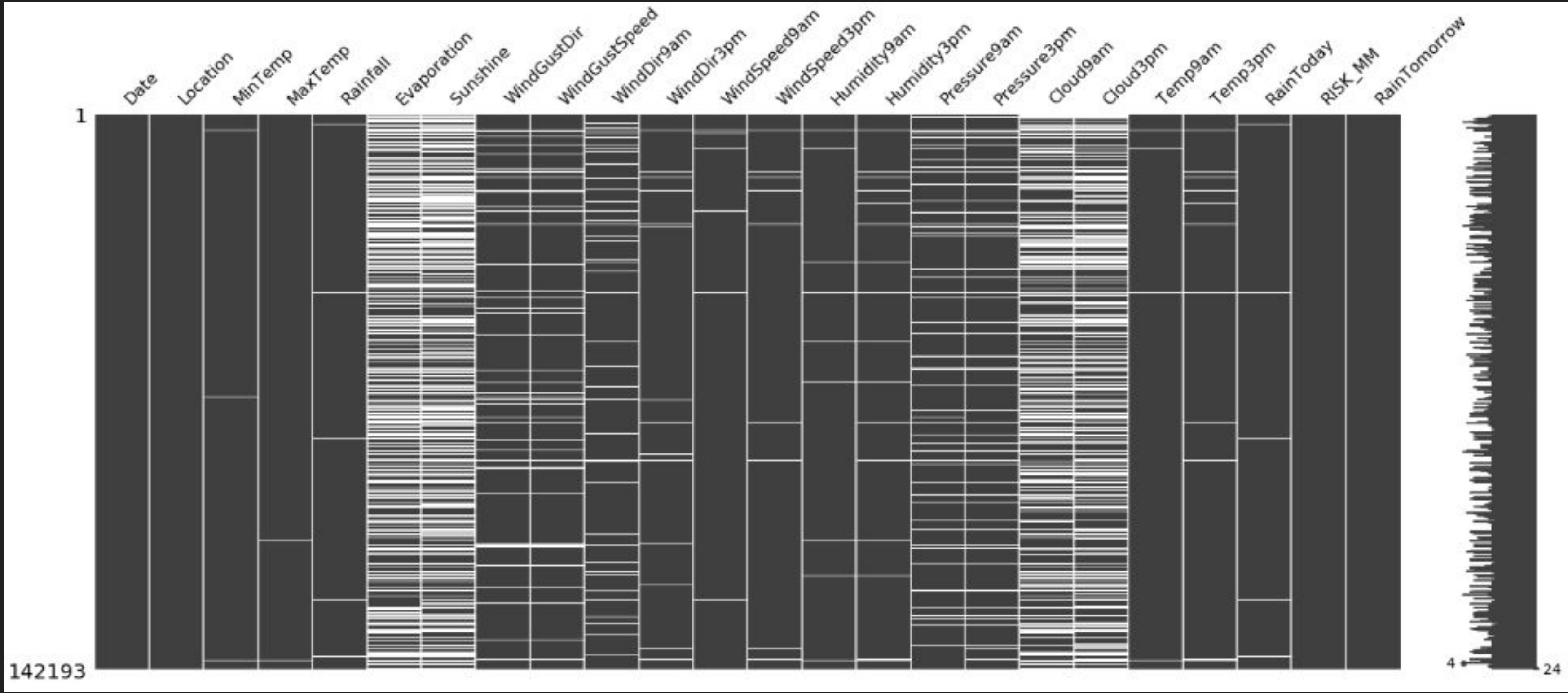
In [66 – 72] – Hyper Parameter Tuning – GBM

In [67 – 72] – Tuned GBM Model and Evaluation

In [73 – 77] – Model comparison – Logistic Regression

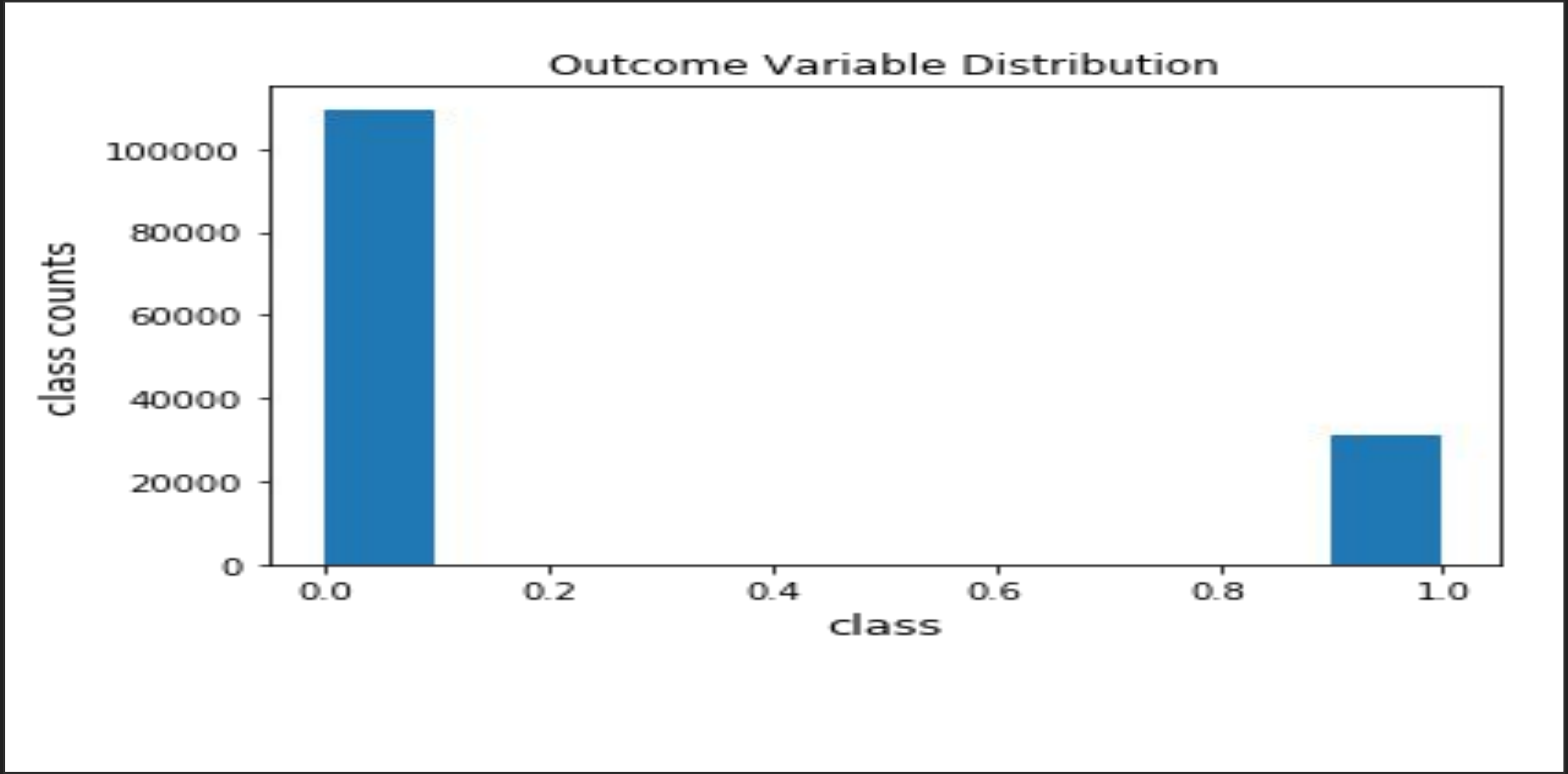
Click here for the GitHub link to the Jupyter notebook

Missing values – white strands indicate missing



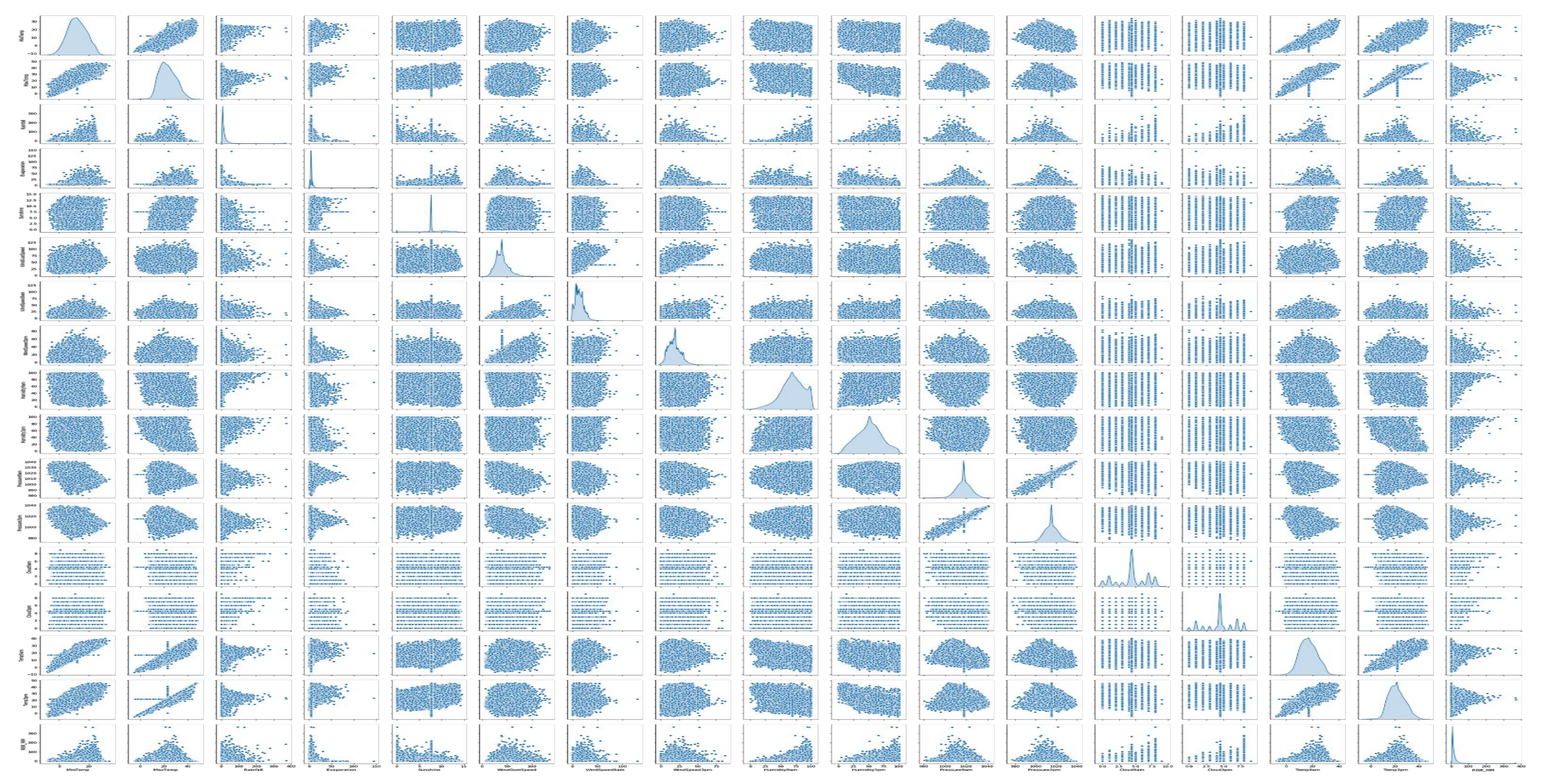
Data shape, before treating NaNs: 142,193 x 24 | Data shape after treating NaNs: 140,787 x 24

Outcome variable – Indicating Class in Balance



‘0.0’ - No Rain Tomorrow, ‘1.0’ - Rain Tomorrow

Bivariate Feature Visualization



ROC/AUC - Comparison