

OSDA. Project report.

Nikolic Stefan

15 December 2019

Introduction

This project focuses on realization of so called lazy classification algorithms from the field of Formal Concept Analysis (FCA). Given a dataset I designed a basic lazy binary classification algorithm with three modifications and tested it on dataset by computing four different classification quality metrics. In order to have something to compare with I also ran three classic classification algorithms on the dataset such as Logistic Regression, SVM and Naive Bayes.

Dataset

In this project I used "Rain in Australia" dataset which contains daily weather observations from numerous Australian weather stations from Kaggle. The target attribute was "Rain-Tomorrow".

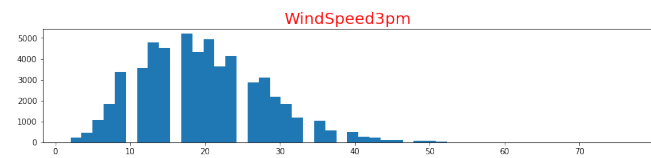
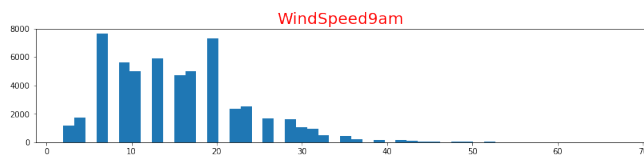
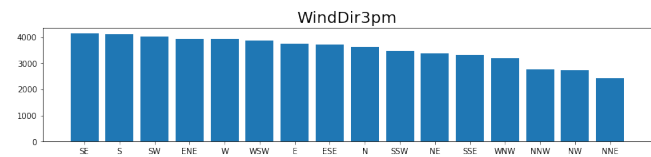
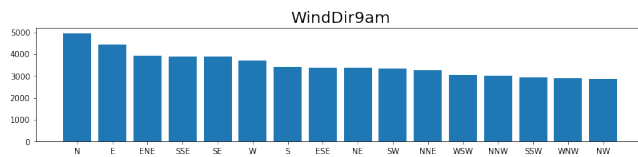
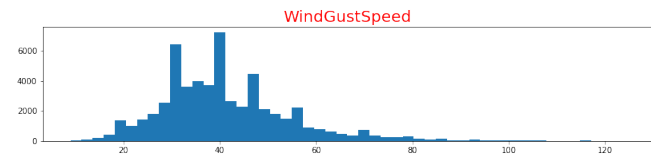
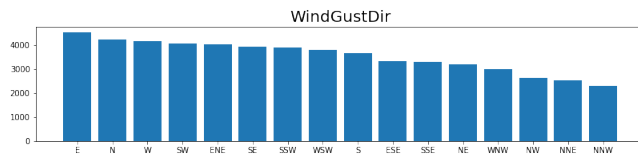
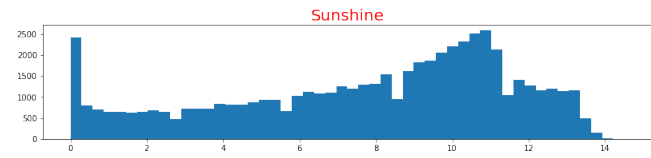
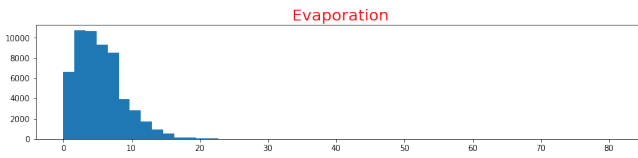
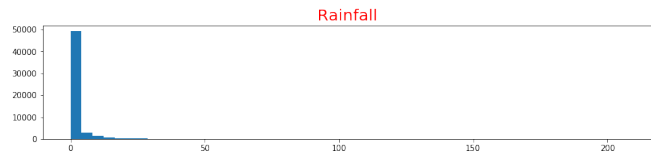
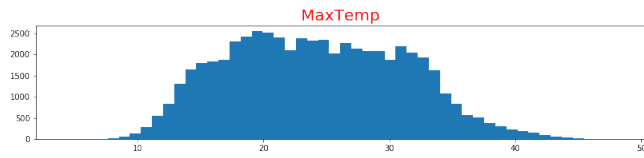
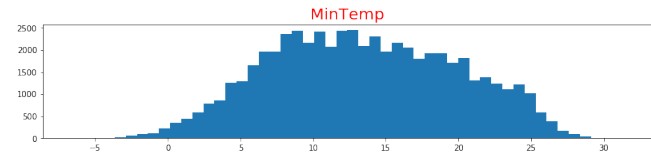
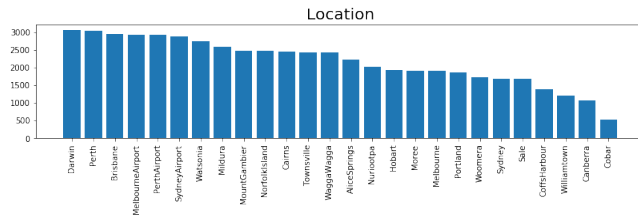
Data contained 24 attributes and 56420 observations (objects). Before starting the exploration of the data I removed attributes "Date" and "RISK_MM" since the first one didn't help and the second one leaks the answers to the model and reduces its predictability.

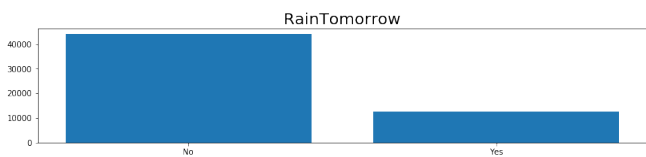
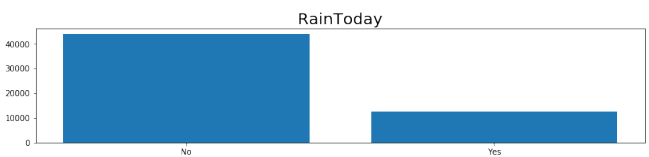
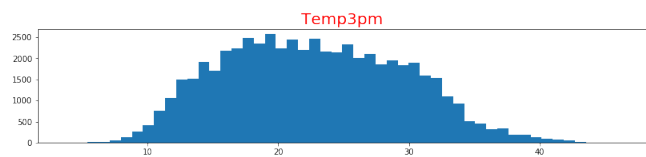
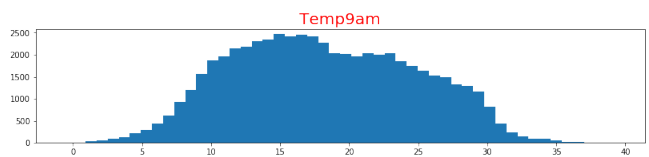
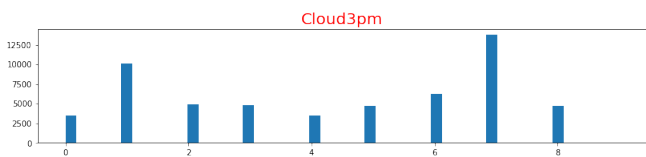
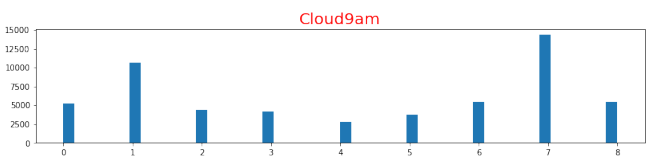
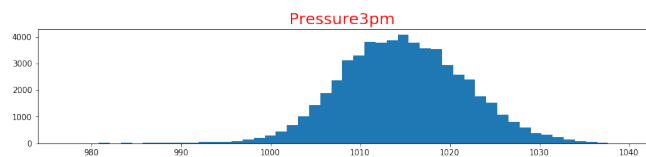
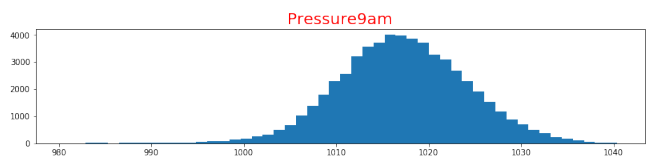
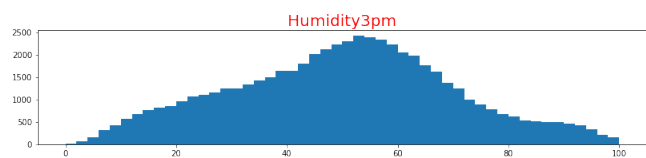
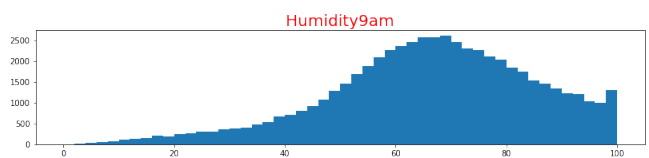
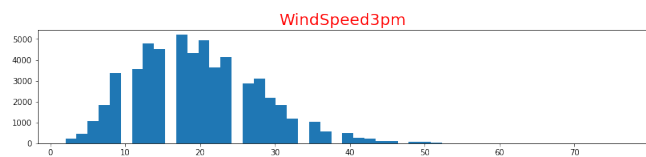
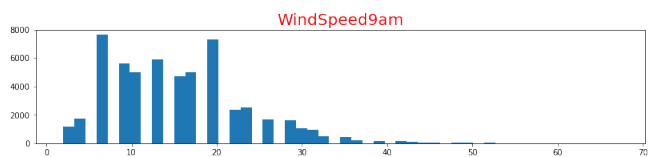
After that, data contained 16 numerical features and 22 categorical ones. Here is the list of all attributes with explicit descriptions:

1. Date: the date of observation
2. Location: The common name of the location of the weather station
3. MinTemp: The minimum temperature in degrees celsius
4. MaxTemp: The maximum temperature in degrees celsius
5. Rainfall: The amount of rainfall recorded for the day in mm
6. Evaporation: The so-called Class A pan evaporation (mm) in the 24 hours to 9am

7. Sunshine: The number of hours of bright sunshine in the day.
8. WindGustDir: The direction of the strongest wind gust in the 24 hours to midnight
9. WindGustSpeed: The speed (km/h) of the strongest wind gust in the 24 hours to midnight
10. WindDir9am: Direction of the wind at 9am
11. WindDir3pm: Direction of the wind at 3pm
12. WindSpeed9am: Wind speed (km/hr) averaged over 10 minutes prior to 9am
13. WindSpeed3pm: Wind speed (km/hr) averaged over 10 minutes prior to 3pm
14. Humidity9am: Humidity (percent) at 9am
15. Humidity3pm: Humidity (percent) at 3pm
16. Pressure9am: Atmospheric pressure (hpa) reduced to mean sea level at 9am
17. Pressure3pm: Atmospheric pressure (hpa) reduced to mean sea level at 3pm
18. Cloud9am: Fraction of sky obscured by cloud at 9am. This is measured in "oktas", which are a unit of eighths. It records how many eighths of the sky are obscured by cloud. A 0 measure indicates completely clear sky whilst an 8 indicates that it is completely overcast.
19. Cloud3pm: Fraction of sky obscured by cloud (in "oktas": eighths) at 3pm. See Cloud9am for a description of the values
20. Temp9am: Temperature (degrees C) at 9am
21. Temp3pm: Temperature (degrees C) at 3pm
22. RainToday: Boolean 1 if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0
23. RainTomorrow: The target variable. Did it rain tomorrow?

Next, I provide some graphic illustration on the distribution of attributes values in the dataset. The red attribute names means that the feature is numerical, otherwise it's categorical:





Binarization

As we know, In order for FCA lazy classification algorithm to work on some dataset it must contain only binary features. Thus, here I use some techniques to make this binarization:

- First of all, I did one-hot encoding on categorical features. This technique transformed all of my categorical features into binary ones, however it increased the number of attributes.
- For numerical I used two strategies. First one is to transform given numerical feature into two by applying so called Gini Split technique as it's done in decision trees.

In order to split given attribute column in 2 columns I try to maximize this target function:

$$Q(R, j) = H(R) - \frac{|R_0|}{|R|}H(R_0) - \frac{|R_1|}{|R|}H(R_1)$$

Where, $H(R) = 1 - \sum_{k=0}^1 p_k^2$ is Gini Index and R is a given column, p_k - is a proportion of objects of class k , R_0 and R_1 two new columns after splitting R by threshold j (all objects $\leq j$ went to column R_0 with value 1 and 0 otherwise).

To choose optimal threshold I maximize my target function Q .

- The second strategy for numerical attribute binarization is to split each numerical column in to three by using quantile intervals.

First column value is 1 if it's less than 0.33 quantile and zero otherwise, second column value is 1 if it's between 0.33 and 0.66 quantiles and zero otherwise and the third column value is 1 if it's value is above 0.66 quantile and zero otherwise.

After the process of the binarization due to having two strategies for numerical attributes I ended up with two binary datasets "quantile_split_data" and "gini_split_data". Lazy classification algorithms were tested on them and baseline algorithms were tested on one-hot encoded data but with untouched numerical attributes.

Lazy classification

First of all, I would like to note that due to computational and runtime difficulties I opted to significantly cut my datasets "quantile_split_data" and "gini_split_data" down to 1000 objects (random selection while maintaining the balance between the classes 500 "+" and 500 "-").

To evaluate all of the algorithms I used standard sklearn train and test split with proportion 0.8 to 0.2.

The first algorithm is as follows:

1. For unclassified object x I intersect it with all of the positive examples $g^+ \in G^+$ and all the negative examples $g^- \in G^-$. After each intersection $x \cap g^{+,-}$ I take its cardinality $|x \cap g^{+,-}|$ and compare it with the threshold t I optimize later for this algorithm.
2. If $|x \cap g^+| \geq t$ I increment the positive votes $vote_pos$ for object x and if $|x \cap g^-| \geq t$ I increment the negative votes $vote_neg$ for object x .
3. To decide the class of x I compare $vote_pos$ and $vote_neg$: if $vote_pos \geq vote_neg$ I label x as "+" and "-" otherwise.

The best results on "gini_split_data":

	Lazy Classifier 1
accuracy score	0.685
precision score	0.693
recall score	0.686
f1 score	0.689

The best results on "quantile_split_data":

	Lazy Classifier 1
accuracy score	0.745
precision score	0.754
recall score	0.788
f1 score	0.771

The second algorithm is as follows:

1. For unclassified object x I intersect it with all of the positive examples $g^+ \in G^+$ and all the negative examples $g^- \in G^-$. After each intersection $x \cap g^{+,-}$ I compute its cardinality $|x \cap g^{+,-}|$ and add it to $intersection_neg$ or $intersection_pos$ variable depending on the class.

2. If $|x \cap g^+| \geq t$ I increment the positive votes $vote_pos$ for object x and if $|x \cap g^-| \geq t$ I increment the negative votes $vote_neg$ for object x .
3. To decide the class of x I compute the following formulas:

$$vote^- = \left(\frac{vote_neg}{|G^-|} + 1 \right) \cdot \frac{intersection_neg}{N_{att}}$$

$$vote^+ = \left(\frac{vote_pos}{|G^+|} + 1 \right) \cdot \frac{intersection_pos}{N_{att}}$$

4. Then, x is classified as "+" if $vote^+ \geq vote^-$ or as "-" otherwise. Parameter t again, is optimized in code.

The best results on "gini_split_data":

	Lazy Classifier 2
accuracy score	0.63
precision score	0.733
recall score	0.431
f1 score	0.543

The best results on "quantile_split_data":

	Lazy Classifier 2
accuracy score	0.72
precision score	0.662
recall score	0.99
f1 score	0.794

The third algorithm is as follows (simpler version of the second):

1. For unclassified object x I intersect it with all of the positive examples $g^+ \in G^+$ and all the negative examples $g^- \in G^-$. After each intersection $x \cap g^{+, -}$ I take it's cardinality $|x \cap g^{+, -}|$ and add it to $intersection_neg$ or $intersection_pos$ variable depending on the class.
2. To decide the class of x I compute the following formulas:

$$vote^- = \frac{intersection_neg}{N_{att}}$$

$$vote^+ = \frac{intersection_pos}{N_{att}}$$

3. To decide the class of x I compare $agg = \frac{vote_{pos}}{vote_{neg}}$ with optimized threshold t : if $agg \geq t$ I label x as "+" and otherwise as "-".

The best results on "gini_split_data":

	Lazy Classifier 3
accuracy score	0.595
precision score	0.672
recall score	0.401
f1 score	0.503

The best results on "quantile_split_data":

	Lazy Classifier 3
accuracy score	0.61
precision score	0.582
recall score	1
f1 score	0.736

Baseline algorithms

Below I provide the results on the baseline algorithms performance, again I tested them on randomly selected subset of 1000 objects (balanced by classes) of my data:

	Logistic Regression	SVM	Naive Bayes
accuracy score	0.725	0.72	0.67
precision score	0.742	0.693	0.672
recall score	0.721	0.826	0.711
f1 score	0.731	0.754	0.691

Conclusion

According to the results I think that lazy classification algorithms are good for scenarios with small dataset and binary nature of the data due to them being easy to apply and having good logical interpretation ability.

As we can see the performance of the first algorithm on "quantile_splitdata" is beating baselines algorithms on almost all metrics.

However, unfortunately, I think lazy classification suffers with two big limitations: it's hard to apply on the data of non-binary nature and it's computational complexity grows exponentially with the size of the data.