AUSTRALIA RAIN PREDICTION

Daniele Petrillo
Maria Zampella

Dataset

The Rain Australia dataset contains information about daily weather observations from many location across Australia. The goal of any machine learning algorithm trained on this dataset is to predict whether it's going to rain during the next day, given information about certain parameters collected during the current day (and, possibly, the previous ones).

The dataset contains about 10 years of daily weather observations.
RainTomorrow is the target variable to predict. It means -- did it rain the next day, Yes or No? This column is Yes if the rain for that day was 1mm or more.

16 numerical features:

MinTemp, MaxTemp, Rainfall, WindGustSpeed, WindSpeed9am, WindSpeed3pm, Humidity9am, Humidity3pm, Pressure9am, Pressure3pm, Temp9am, Temp3pm, Evaporation, Sunshine, Cloud9am, Cloud3pm.

6 categorical features:

Date, Location, WindGustDir, WindDir9am,WindDir3pm, RainToday.

The target variable: RainTomorrow

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RainTomorrow
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	71.0	22.0	1007.7	1007.1	8.0	NaN	16.9	21.8	No	No
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	44.0	25.0	1010.6	1007.8	NaN	NaN	17.2	24.3	No	No
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	38.0	30.0	1007.6	1008.7	NaN	2.0	21.0	23.2	No	No
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	45.0	16.0	1017.6	1012.8	NaN	NaN	18.1	26.5	No	No
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	82.0	33.0	1010.8	1006.0	7.0	8.0	17.8	29.7	No	No
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
145455	2017-06-21	Uluru	2.8	23.4	0.0	NaN	NaN	E	31.0	SE	...	51.0	24.0	1024.6	1020.3	NaN	NaN	10.1	22.4	No	No
145456	2017-06-22	Uluru	3.6	25.3	0.0	NaN	NaN	NNW	22.0	SE	...	56.0	21.0	1023.5	1019.1	NaN	NaN	10.9	24.5	No	No
145457	2017-06-23	Uluru	5.4	26.9	0.0	NaN	NaN	N	37.0	SE	...	53.0	24.0	1021.0	1016.8	NaN	NaN	12.5	26.1	No	No
145458	2017-06-24	Uluru	7.8	27.0	0.0	NaN	NaN	SE	28.0	SSE	...	51.0	24.0	1019.4	1016.5	3.0	2.0	15.1	26.0	No	No
145459	2017-06-25	Uluru	14.9	NaN	0.0	NaN	NaN	NaN	NaN	ESE	...	62.0	36.0	1020.2	1017.9	8.0	8.0	15.0	20.9	No	NaN

145460 rows × 23 columns

WORKFLOW

Preprocessing

01

- Missing data handling
- Correlation
- Categorical data encoding
- Rolling averages

Dimensionality reduction

02

Feature selection:

- Variance Analysis
- Random Forest Feature Importance

Feature extraction:

- PCA
- LDA

Model application:

03

- Perceptron
- Logistic Regression
- Random Forest
- Ensamble method
- Neural Network

PREPROCESSING

Missing data removal

01 Remove columns with high percentage on NaN

02 Remove data with NaN for RainTomorrow column.

Column	N° of Nan	% of Nan
Date	0	0.0
Location	0	0.0
MinTemp	1485	1.02
MaxTemp	1261	0.87
Rainfall	3261	2.24
Evaporation	62790	43.17
Sunshine	69835	48.01
WindGustDir	10326	7.1
WindGustSpeed	10263	7.06
WindDir9am	10566	7.26
WindDir3pm	4228	2.91
WindSpeed9am	1767	1.21
WindSpeed3pm	3062	2.11
Humidity9am	2654	1.82
Humidity3pm	4507	3.1
Pressure9am	15065	10.36
Pressure3pm	15028	10.33
Cloud9am	55888	38.42
Cloud3pm	59358	40.81
Temp9am	1767	1.21
Temp3pm	3609	2.48
RainToday	3261	2.24
RainTomorrow	3267	2.25

PREPROCESSING

Missing data removal

03 Remove all the istances of a Location which has any column with all Null values

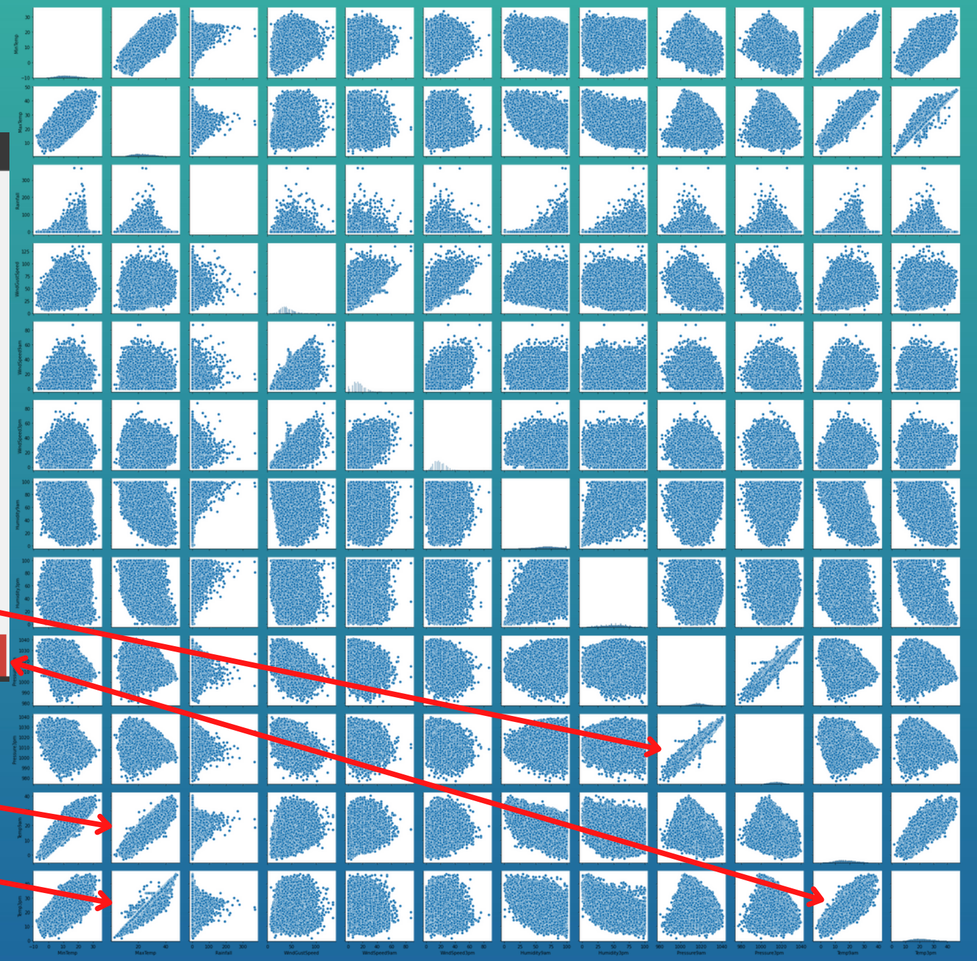
04 Substitute the Null values with the mean/mode of the specific column grouped by location

Location	MinTemp	MaxTemp	Rainfall	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Temp9am	Temp3pm
Adelaide	12.628368	22.945402	1.572185	36.530812	9.954295	15.470665	59.618476	44.820097	1018.727579	1016.772202	16.973193	21.603953
Albany	12.948461	20.072587	2.255073	NaN	12.498986	19.001305	74.820202	67.392487	1018.283049	1016.500668	16.233154	18.412974
Albury	9.520899	22.630963	1.925710	32.953016	8.221816	14.378828	74.108081	47.884935	1018.367253	1015.755504	14.348620	21.364716
AliceSprings	13.125182	29.244191	0.869355	40.533714	14.728623	18.103665	39.625165	24.078321	1016.699670	1012.884478	21.328868	28.008452
MountGinini	3.651193	11.777947	3.245241	46.188214	15.922178	15.128915	77.131725	69.683854	NaN	NaN	6.890845	10.277890
Newcastle	13.740240	24.098283	3.153022	NaN	5.985150	10.623558	74.011532	57.345455	NaN	NaN	18.102842	22.316698
Nhil	8.992798	22.398407	0.932907	42.542438	16.443595	20.956023	73.038241	44.924793	1018.546335	1016.401402	13.377693	20.866348
NorahHead	15.375197	22.607900	3.382479	42.215043	13.912775	21.020240	74.610577	67.538937	1018.259033	1016.014824	18.489389	20.781722
NorfolkIsland	16.839960	21.792746	3.137568	42.639973	20.168471	21.892640	70.810071	67.840648	1017.631360	1015.887576	19.745353	20.436158
Nuriootpa	9.366433	21.714658	1.381375	40.660720	14.237345	18.166275	66.724265	45.366343	1018.915793	1016.791316	14.858992	20.319098
PearceRAAF	12.386314	26.271049	1.634354	43.448549	16.029872	20.261409	60.330783	41.338568	1017.746902	1015.177575	18.961631	24.740635
Penrith	12.533649	24.731984	2.135143	31.794386	6.343878	11.968168	75.787124	48.955953	NaN	NaN	16.949406	23.334496

PREPROCESSING

Correlation

	MinTemp	MaxTemp	Rainfall	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Temp9am
MinTemp											
MaxTemp	0.73										
Rainfall	0.11	-0.07									
WindGustSpeed	0.20	0.09	0.13								
WindSpeed9am	0.20	0.03	0.09	0.60							
WindSpeed3pm	0.17	0.04	0.06	0.68	0.51						
Humidity9am	-0.24	-0.52	0.22	-0.22	-0.27	-0.13					
Humidity3pm	0.02	-0.49	0.25	-0.04	-0.03	0.03	0.67				
Pressure9am	-0.45	-0.33	-0.17	-0.45	-0.23	-0.30	0.14	-0.02			
Pressure3pm	-0.46	-0.42	-0.13	-0.40	-0.18	-0.26	0.19	0.05	0.96		
Temp9am	0.90	0.88	0.02	0.17	0.15	0.16	-0.48	-0.20	-0.42	-0.47	
Temp3pm	0.71	0.98	-0.07	0.05	0.02	0.02	-0.51	-0.54	-0.29	-0.39	0.86



05 Remove column which have an high correlation:
Temp9am, Temp3pm, Pressure9am

PREPROCESSING

Data column handling

01 Extract month information from data column.

02 Build dummy variables for each month.

Categorical data encoding

03 Build dummy variables for each categorical column.

04 Binary encoding for RainToday and RainTomorrow columns.

```
# extract the month of the year from the string date

temp = pd.read_csv("gdrive/MyDrive/mlxp/df_filled.csv")
df_date = temp[["Date"]]
months = []
for date in df_date["Date"]:
  month = int(date.split(sep="-")[1])
  months.append(month)

# substitute all the new values
df_date["Date"] = months
df_date.rename(columns={"Date": "Month"}, inplace=True)
df_date
```

	Month
0	12
1	12
2	12
3	12
4	12
...	...
127391	6
127392	6
127393	6
127394	6
127395	6

127396 rows × 1 columns

December	February	March	April	May	June
1	0	0	0	0	0
1	0	0	0	0	0
1	0	0	0	0	0
1	0	0	0	0	0
1	0	0	0	0	0
...	...	...	...	...	...
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1

The encoded categorical variables look like this:

	RainToday	RainTomorrow
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
...	...	...
127391	0	0
127392	0	0
127393	0	0
127394	0	0
127395	0	0

127396 rows × 2 columns

	Albury	AliceSprings	BadgerysCreek	Ballarat	Bendigo	Brisbane	Cairns	Canberra	Cobar	CoffsHarbour	...
0	1	0	0	0	0	0	0	0	0	0	...
1	1	0	0	0	0	0	0	0	0	0	...
2	1	0	0	0	0	0	0	0	0	0	...
3	1	0	0	0	0	0	0	0	0	0	...
4	1	0	0	0	0	0	0	0	0	0	...
...	...	...	...	...	...	...	...	...	...	...	...
127391	0	0	0	0	0	0	0	0	0	0	...
127392	0	0	0	0	0	0	0	0	0	0	...
127393	0	0	0	0	0	0	0	0	0	0	...
127394	0	0	0	0	0	0	0	0	0	0	...
127395	0	0	0	0	0	0	0	0	0	0	...

127396 rows × 43 columns

	WindGustDir_ENE	WindGustDir_ESE	WindGustDir_N	WindGustDir_NE	WindGustDir_NNE	WindGustDir_NNW	WindGustDir_NW	WindGustDir_S	WindGustDir_SE	WindGustDir_SSE	...
0	0	0	0	0	0	0	0	0	0	0	...
1	0	0	0	0	0	0	0	0	0	0	...
2	0	0	0	0	0	0	0	0	0	0	...
3	0	0	0	1	0	0	0	0	0	0	...
4	0	0	0	0	0	0	0	0	0	0	...
...	...	...	...	...	...	...	...	...	...	...	...
127391	0	0	0	0	0	0	0	0	0	0	...
127392	0	0	0	0	0	0	0	0	0	0	...
127393	0	0	0	0	0	1	0	0	0	0	...
127394	0	0	1	0	0	0	0	0	0	0	...
127395	0	0	0	0	0	0	0	0	1	0	...

127396 rows × 45 columns

PREPROCESSING

Rolling averages

The dataset is a time series. To exploit the time aspect we compute the rolling mean of the numerical columns referred to the 3 previous observation (grouped by location).

	MinTemp	MaxTemp	Rainfall	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure3pm	avgMinTemp	avgMaxTemp	avgRainfall	avg
0	13.4	22.9	0.6	44.0	20.0	24.0	71.0	22.0	1007.1	13.400000	22.900000	0.600000	
1	7.4	25.1	0.0	44.0	4.0	22.0	44.0	25.0	1007.8	10.400000	24.000000	0.300000	
2	12.9	25.7	0.0	46.0	19.0	26.0	38.0	30.0	1008.7	11.233333	24.566667	0.200000	
3	9.2	28.0	0.0	24.0	11.0	9.0	45.0	16.0	1012.8	9.833333	26.266667	0.000000	
4	17.5	32.3	1.0	41.0	7.0	20.0	82.0	33.0	1006.0	13.200000	28.666667	0.333333	
...	...	...	...	...	...	...	...	...	...	...	...	...	
127391	3.5	21.8	0.0	31.0	15.0	13.0	59.0	27.0	1021.2	6.300000	21.033333	0.000000	
127392	2.8	23.4	0.0	31.0	13.0	11.0	51.0	24.0	1020.3	4.566667	21.933333	0.000000	
127393	3.6	25.3	0.0	22.0	13.0	9.0	56.0	21.0	1019.1	3.300000	23.500000	0.000000	
127394	5.4	26.9	0.0	37.0	9.0	9.0	53.0	24.0	1016.8	3.933333	25.200000	0.000000	
127395	7.8	27.0	0.0	28.0	13.0	7.0	51.0	24.0	1016.5	5.600000	26.400000	0.000000	

127396 rows × 18 columns

DIMENSIONALITY REDUCTION

Feature Selection: VARIANCE ANALYSIS

Variance is a measure of dispersion of data points from the mean.
Low variance indicates that data points are generally similar
and do not vary widely from the mean.

A baseline approach for feature
selection is to remove all features whose
variance is lower than a certain
threshold.

$$\sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n}$$

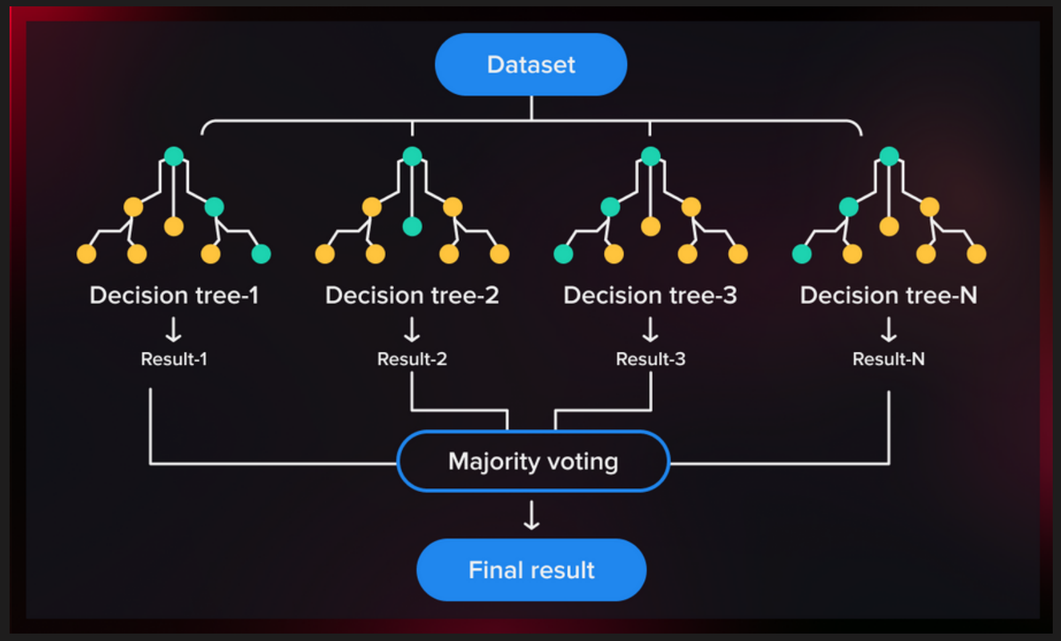
```
sel = VarianceThreshold(threshold = 0.2)
sel.fit_transform(X_train)
print('The numerical columns are: ', columns)
print('The features to keep are', sel.get_feature_names_out(columns))
```

```
The features to keep are ['MinTemp' 'MaxTemp' 'Rainfall' 'WindGustSpeed' 'WindSpeed9am'
 'WindSpeed3pm' 'Humidity9am' 'Humidity3pm' 'Pressure3pm' 'avgMinTemp'
 'avgMaxTemp' 'avgRainfall' 'avgWindGustSpeed' 'avgWindSpeed9am'
 'avgWindSpeed3pm' 'avgHumidity9am' 'avgHumidity3pm' 'avgPressure3pm']
```

DIMENSIONALITY REDUCTION

Feature Selection: RANDOM FOREST FEATURE IMPORTANCE

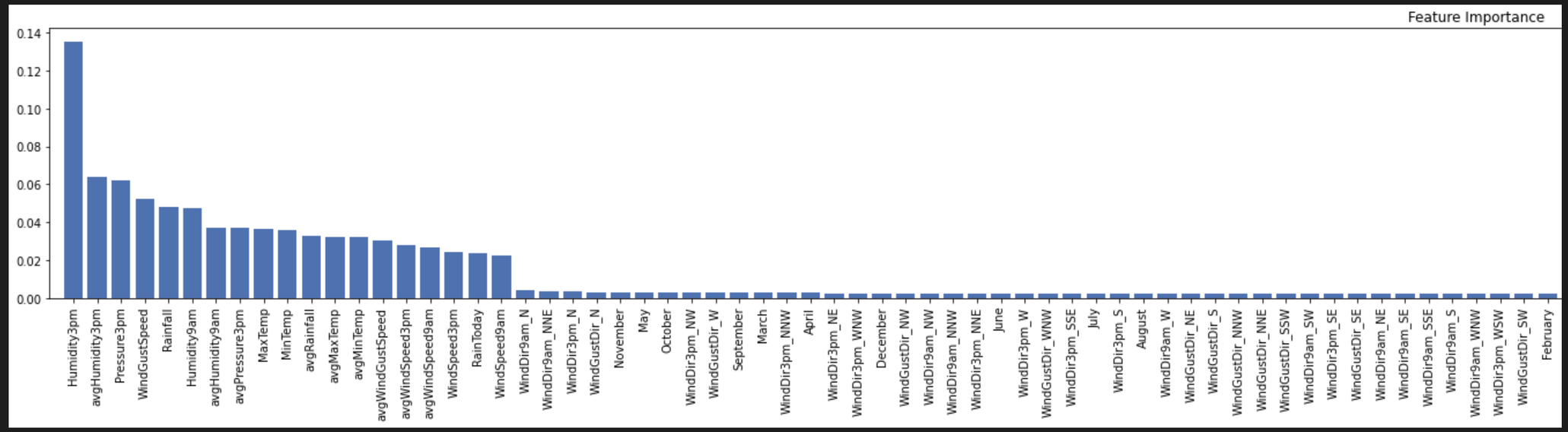
Using a random forest, we can measure the feature importance as the averaged impurity decrease computed from all decision trees in the forest, without making any assumptions about whether our data is linearly separable or not.

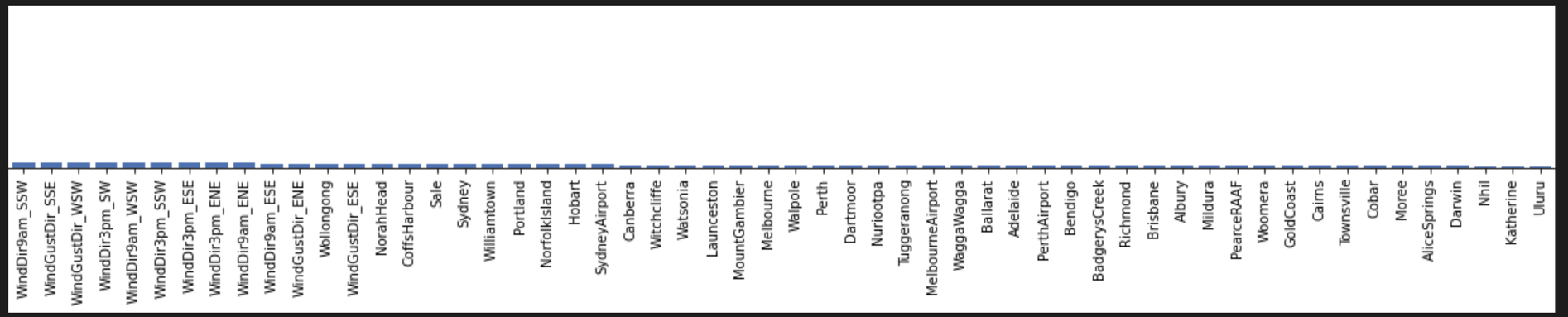


To give a better intuition, features that are selected at the top of the trees are in general more important than features that are selected at the end nodes of the trees, as generally the top splits lead to bigger information gains.

DIMENSIONALITY REDUCTION

Feature Selection: RANDOM FOREST FEATURE IMPORTANCE





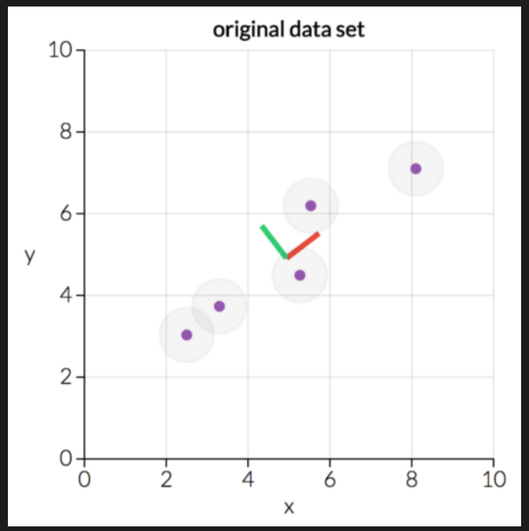
DIMENSIONALITY REDUCTION

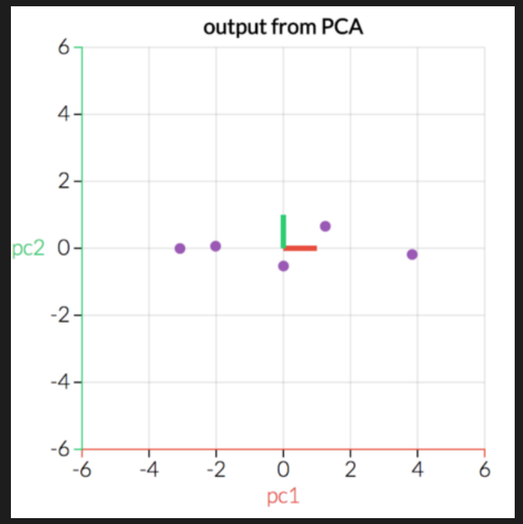
Feature Extraction: PCA

PCA is an approach based on an unsupervised linear transofrmation technique.

The basic idea is that PCA finds the directions of maximum variance in data and project it onto a new subspace with equal of fewer dimensions than the original one (the dimensions are called principal components).

All the component of the new space are uncorrelated, so the principal component are orthogonal.





DIMENSIONALITY REDUCTION

Feature Extraction: PCA

01

- Standardize the d-dimensional dataset

02

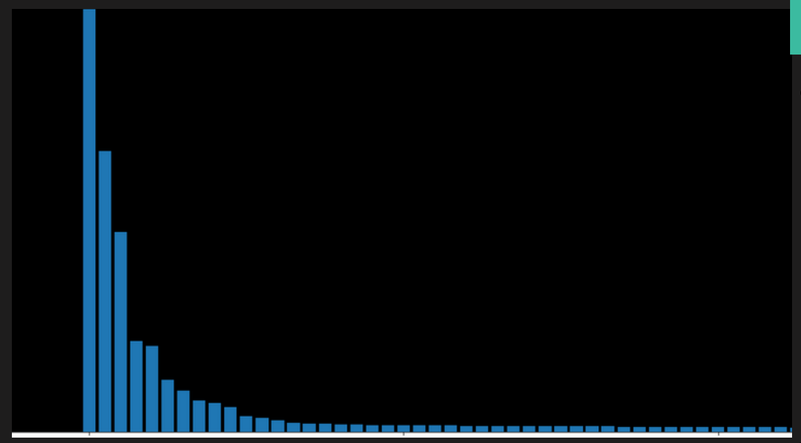
- Costruct the covariance matrix

03

- Decompose the covariance matrix into its eigenvectors and eigenvalues

04

- Sort the eigenvalues by decreasing order to rank the corresponding eigenvectors.



05

- Select k eigenvectors which correspond to the k largest eigenvalues, where k is the dimensionality of the new feature subspace (k <= d).

06

- Construct a projection matrix W from the ”top” k eigenvectors.

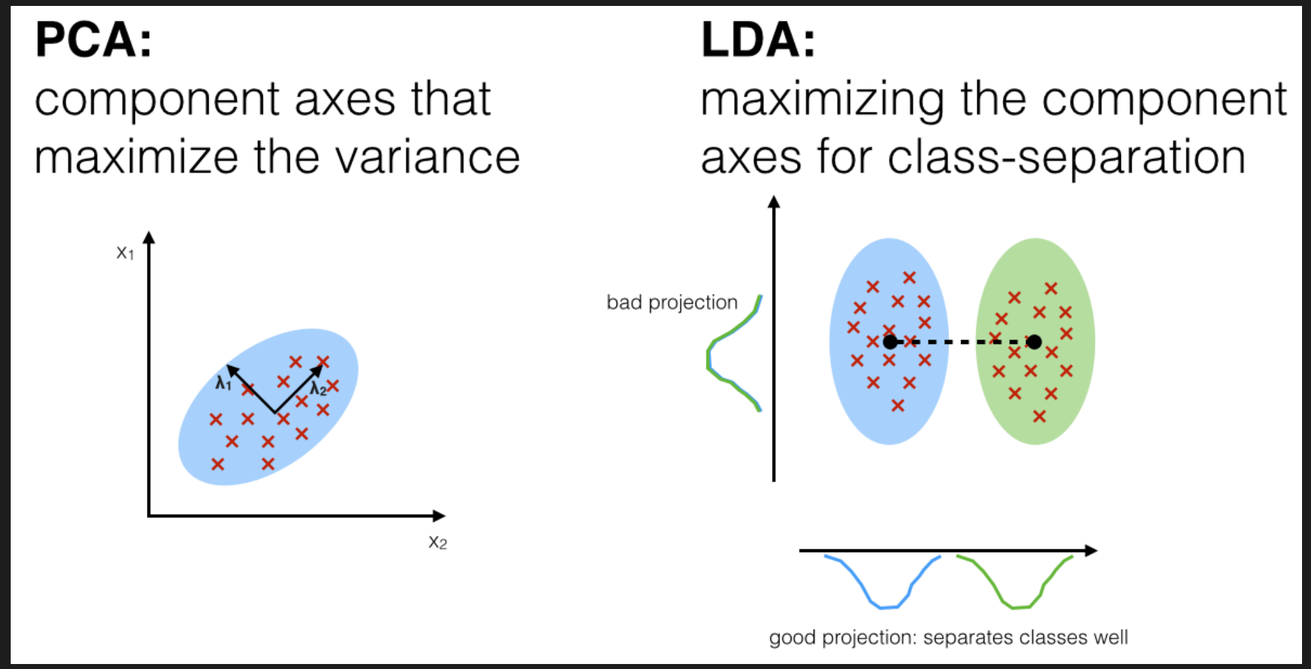
07

- Transform the d-dimensional input dataset X using the projection matrix W to obtain the new k-dimensional feature subspace.

DIMENSIONALITY REDUCTION

Feature Extraction: LDA

Linear Discriminant Analysis (LDA) shares with PCA the ultimate goal: to create a new set of features that better explains the relation between instances and their corresponding target value.



The main difference between the two is the fact that LDA builds the features that better separates the output classes.

DIMENSIONALITY REDUCTION

Feature Extraction: LDA

01

- Standardize the d-dimensional dataset

02

- For each class, compute the d-dimensional mean vector

03

- Construct the between-class scatter matrix SB and the within-class scatter matrix Sw

04

- Compute the eigenvectors and corresponding eigenvalues of the matrix Sw−1SB

05

- Sort the eigenvalues by decreasing order to rank the corresponding eigenvectors

06

- Choose the k eigenvectors that correspond to the k largest eigenvalues to construct a d ⇥ k - dimensional transformation matrix W; the eigenvectors are the columns of this matrix.

07

- Project the samples onto the new feature subspace using W

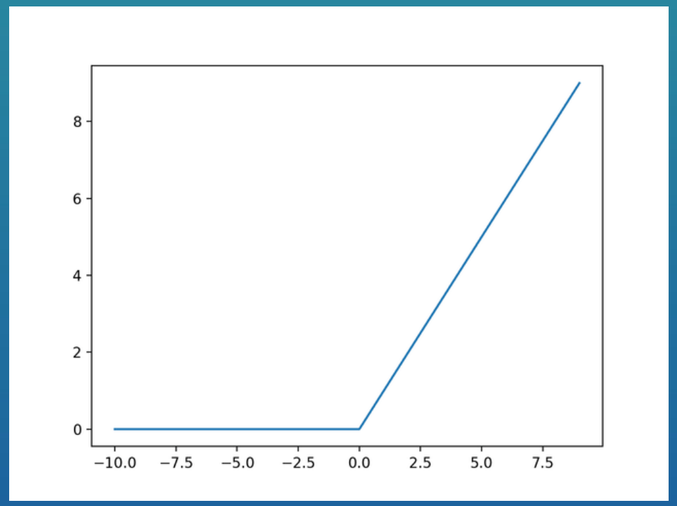
MODEL APPLICATION

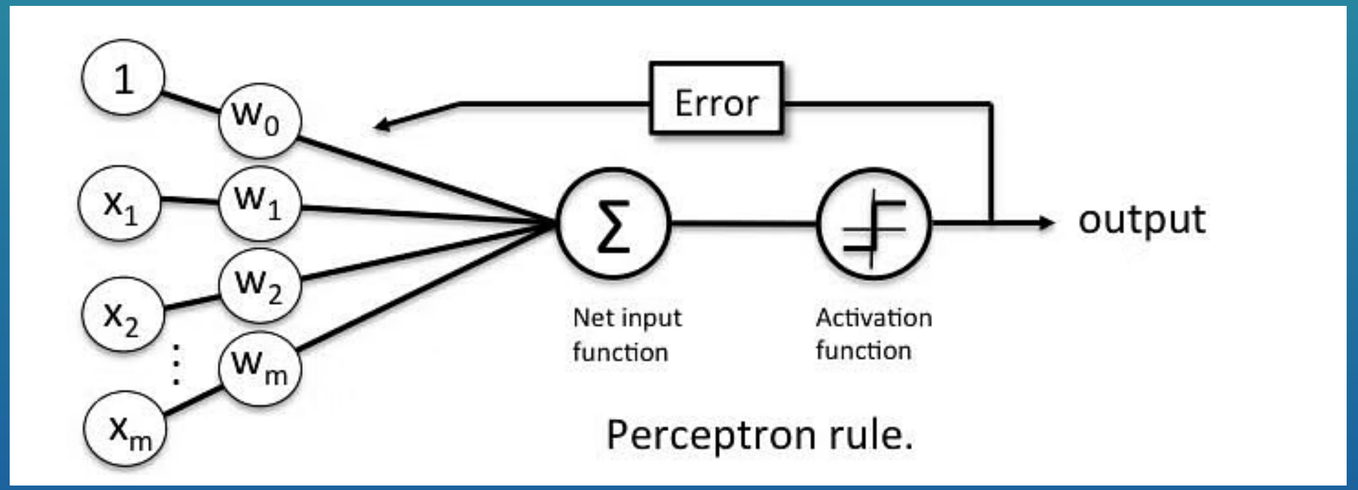
Perceptron

The Perceptron is a specific implementation of an artificial neuron: in particular, what distinguishes it from the orignal formulation is the fact that it "learns" its own weights and biasis by itself.

It uses an activation function, which is applied to the net input of the neuron.

A perceptron "learns" by confronting its guessed output to the real one, then backtracking the error.





MODEL APPLICATION

Logistic Regression

- It is a probabilistic model.
- Its goal is to model the probability that an observation belongs to the positive class.
- The aim of the logistic regression is to compute: $logit(p(y = 1|\mathbf{x}))$

where $logit(p) = \log \frac{p}{(1-p)}$

- The logit function takes input values in [0, 1] and transforms them to values over the entire real number range, which we can use to express a linear relationship between feature values and the log-odds.
- What we are actually interested in is predicting the probability that a certain sample belongs to a particular class, which is the inverse form of the logit function: the sigmoid function:

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

MODEL APPLICATION

Logistic Regression - Regularization

- Regularization is a very useful method to filter out noise from data, and eventually prevent overfitting.

- Regularization introduces additional information to penalize high weights. Thus a setting of the weights that matches the training data perfectly, but uses many weights with high values to do so, will be penalized more than a setting that matches the data a little less well, but does so using smaller weights.

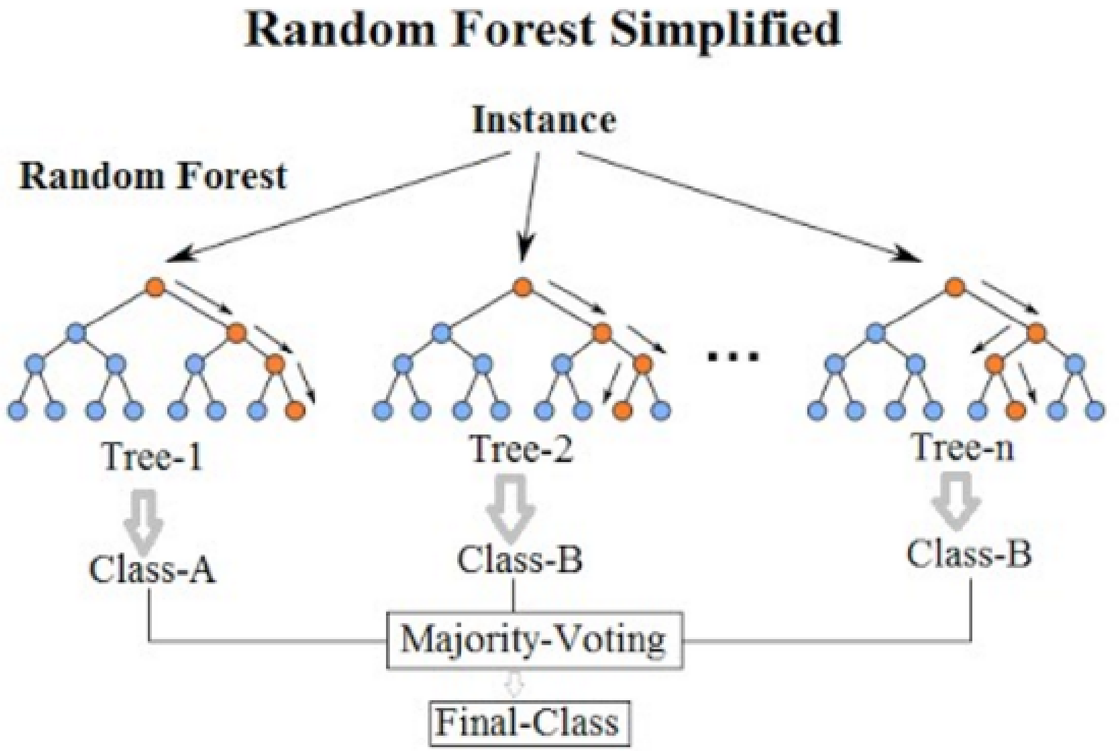
- The most common form of regularization is the so-called L2 regularization:

$$\frac{\lambda}{2}\|\mathbf{w}\|_2^2 = \frac{\lambda}{2}\sum_{j=1}^{m} w_j^2$$

MODEL APPLICATION

Random Forest

- It is an ensamble of decision trees and it is scale invariant.
- Split the nodes using the feature that provides the best split according to the objective function.
- Aggregate the prediction by each tree to assign the class label by majority vote.
- Shrinking the size of the bootstrap samples may increase the randomness of the random forest, and it can help to reduce the effect of overfitting.

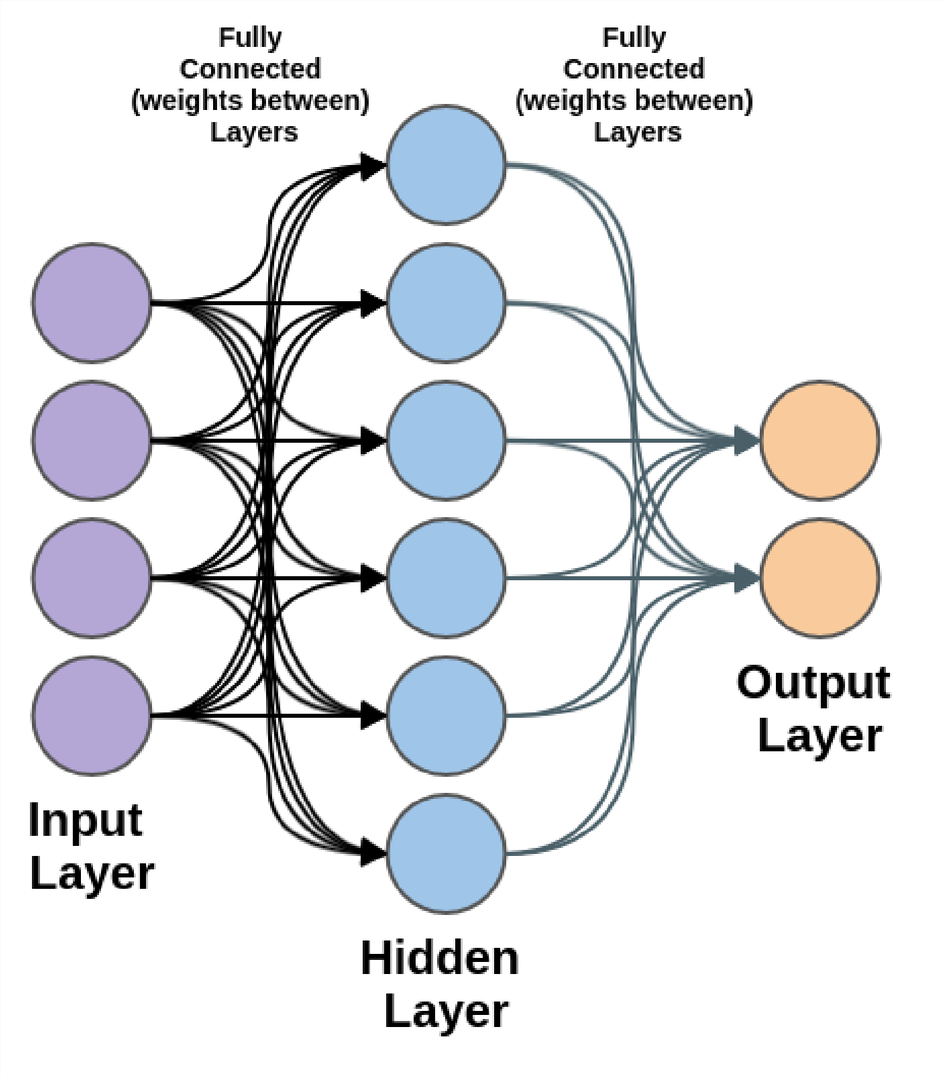


MODEL APPLICATION

Neural Network

A Neural Network is an architecture based on single artificial neurons, which are connected together to form so called "layers".

in particular, a MultiLayer Perceptron (MLP) is a neural network with one input layer, a fixed amount of hidden layers, and one output layer; it is also fully connected, meaning that each neuron of a layer is connected to every neuron of the next one.

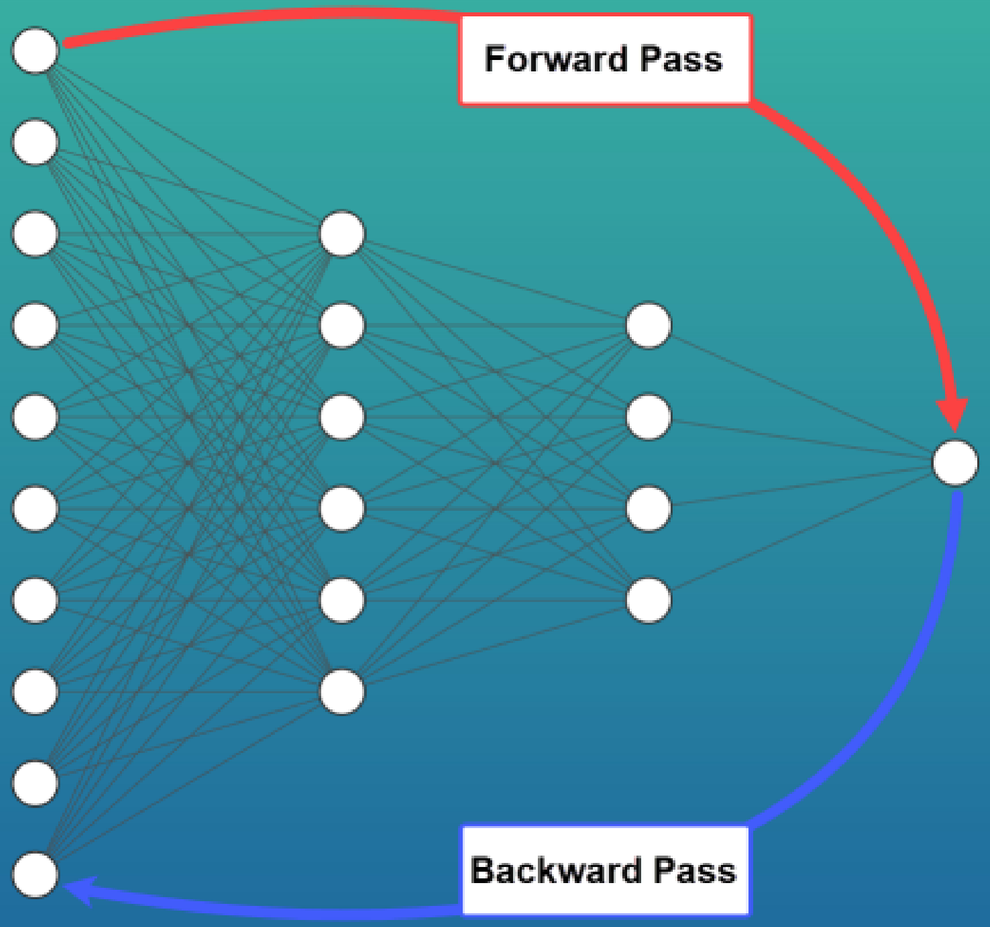


MODEL APPLICATION

Neural Network

The most popular technique to train
a neural network is called BackPropagation:

- First, the output is generated by propagating forward the input.
- Then, the error between the guess and the actual target is calculated.
- Finally, the error is propagated backwards to adjust weights and biases optimally.

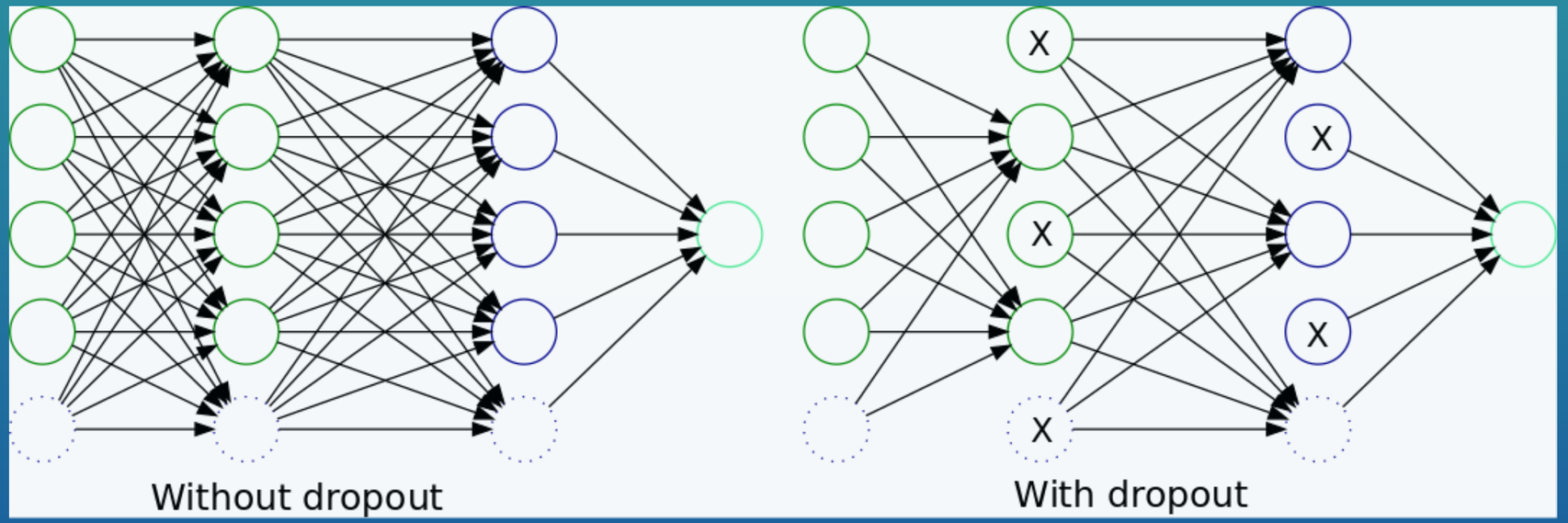


MODEL APPLICATION

Dropout

Sometimes a model can fall in what's called "Overfitting", i.e. it learns too well the training set and doesn't generalize properly to unseen data.

One way to mitigate this problem is to add dropout layers to the network, which randomly "turn off" some neurons.



Our Neural Network

```
# Initialising the NN
model = Sequential()
# layers
model.add(Dense(units = 64,  activation = 'relu', input_dim = X_train_std.shape[1]))
model.add(Dense(units = 128,  activation = 'relu'))
model.add(Dropout(0.3))
model.add(Dense(units = 64,  activation = 'relu'))
model.add(Dropout(0.3))
model.add(Dense(units = 32,  activation = 'relu'))
model.add(Dropout(0.5))
model.add(Dense(units = 1, activation = 'sigmoid'))
# Compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=[keras.metrics.Precision()])
# Fit the model
history = model.fit(X_train_std, y_train, epochs=50, batch_size=32,
                    validation_split=0.2,
                   verbose=1)
#Use the model to predict the label of the test dataset
y_pred_nn = model.predict(X_test_std)>0.5
```

RESULTS

Complete dataset

Perceptron

Best parameter:
eta0: 0.1, penalty= l1

Train Precision:
0.546 ± 0.092
Test precision: 0.483
Test Recall: 0.598
Test F1: 0.534
Test accuracy: 0.768

Misclassified
samples: 5910

Logistic Regression

Best parameter:
C=0.001, sovler='sag'

Train Precision:
0.748±0.013
Test precision: 0.743
Test Recall: 0.482
Test F1: 0.585
Test accuracy: 0.848

Misclassified
samples: 3884

Random Forest

Best parameter:
criterion: entropy,
max_depth:10,
n_est:100

Train Precision:
0.795±0.012
Test precision: 0.786
Test Recall: 0.411
Test F1: 0.540
Test accuracy: 0.844
Misclassified
samples: 3977

Neural Network

Test precision:0.689

Test Recall: 0.561

Test F1: 0.618

Test accuracy: 0.846

Misclassified
samples: 3925

RESULTS

RandomForest dataset

Perceptron

Best parameter:
eta0: 0.1, penalty= l1

Train Precision:
0.546 ± 0.092
Test precision: 0.483
Test Recall: 0.598
Test F1: 0.534
Test accuracy: 0.768

Misclassified
samples: 5910

Logistic Regression

Best parameter:
C=0.001, sovler='sag'

Train Precision:
0.748±0.013
Test precision: 0.743
Test Recall: 0.482
Test F1: 0.585
Test accuracy: 0.848

Misclassified
samples: 3884

Random Forest

Best parameter:
criterion: entropy,
max_depth:10,
n_est:100

Train Precision:
0.795±0.012
Test precision: 0.786
Test Recall: 0.411
Test F1: 0.540
Test accuracy: 0.844
Misclassified
samples: 3977

Neural Network

Test precision:0.689

Test Recall: 0.561

Test F1: 0.618

Test accuracy: 0.846

Misclassified
samples: 3925

RESULTS

Modified dataset: numerical + rolling mean + month dummy + location dummy

Perceptron

Best parameter:
eta0: 0.1, penalty= l1

Train Precision:
0.546 ± 0.091
Test precision: 0.413
Test Recall: 0.477
Test F1: 0443
Test accuracy: 0.732

Misclassified samples: 6817

Logistic Regression

Best parameter:
C=0.001, sovler='saga'

Train Precision:
0.756±0.013
Test precision: 0.755
Test Recall: 0.489
Test F1: 0.549
Test accuracy: 0.851

Misclassified samples: 3795

Random Forest

Best parameter:
criterion: entropy,
max_depth:10,
n_est:100

Train Precision:
0.795±0.013
Test precision: 0.784
Test Recall: 0.415
Test F1: 0.548
Test accuracy: 0.844
Misclassified samples:3967

Neural Network

Test precision:0.736

Test Recall: 0.582

Test F1: 0.650

Test accuracy: 0.860
Misclassified samples: 3555

Conclusions and Future improvements

- The best overall classifier seems to be the Random Forest.
- Perceptron, on the other hand, had the worst performance, probably because of its intrinsic limitations and simplicity.
- Although the scores seem to be quite low, the precision looks promising.
- The dataset may have poorly descriptive features.
- The high cardinality of categorical variables creates some difficulties in feature selection and model performances.

- The unbalanced dataset could be handled in some other ways, e.g. with under/oversampling.
- One possibility to further improve predictions is to implement other models or ensemble methods.
- Some feature selections methods for categorical features could be implemented.
- Instead of a classical Neural Network, a Recurring Neural Network or a Long Short Term Memory could have been used, which better suit time series problems.