CS-618
Deep Learning
LateSpring 2023

Lecture 3

Classification

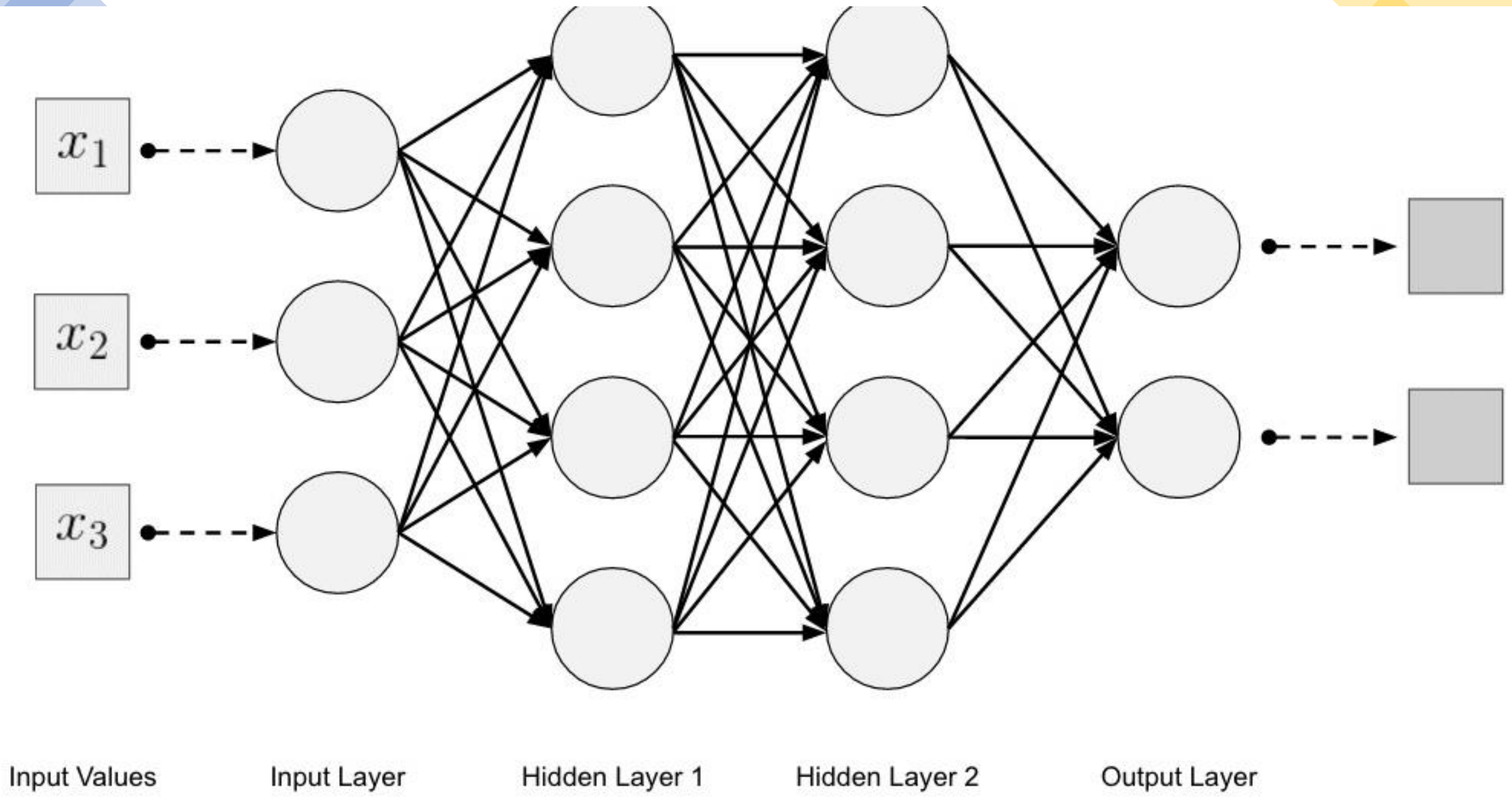
- Classification is modeling based on delineating classes of output based on some set of input features.
- If regression give us an outcome of "*how much*,"

classification gives us an outcome of "*what kind*."

- The dependent variable *y* is categorical rather than numerical.
- binary classifier that only has a single output with two labels
- The output can also be a floating-point number between 0.0 and 1.0
- Examples of binary classification include:
    - Classifying whether someone has a disease or not
    - Classifying an email as spam or not spam
    - Classifying a transaction as fraudulent or nominal
- classification models that have *N* labels

Classification In neural network

- In neural networks, classification refers to the process of categorizing input data into distinct classes or categories. This is typically done by mapping input data to a set of output values that represent the likelihood or probability that the input belongs to each of the predefined classes.
- For example, in image classification, the input data would be an image, and the neural network would be trained to output a set of probabilities that the image belongs to each of the classes, such as "dog," "cat," "bird," etc.
- To achieve this, a neural network typically consists of several layers of interconnected neurons, each of which performs a series of calculations on the input data to transform it into a format that is better suited for classification. These layers are often referred to as the input layer, hidden layers, and output layer.
- During the training process, the neural network adjusts the weights and biases of the neurons to minimize the error between the predicted output and the actual output. Once the network is trained, it can be used to classify new input data by forwarding it through the network and producing an output that indicates the most likely class

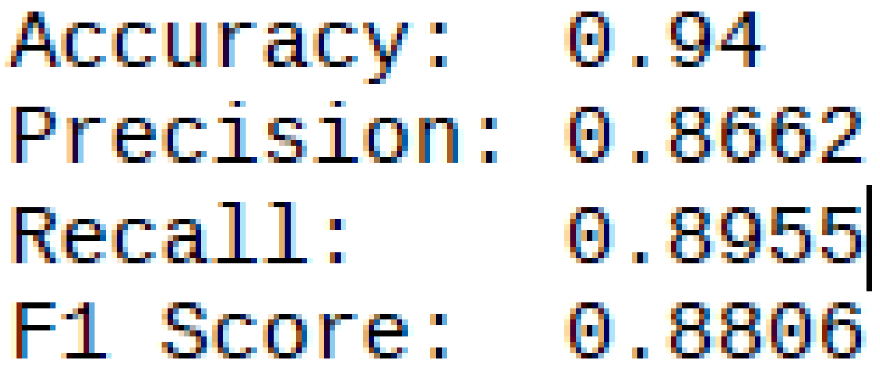

x_1
x_2
x_3
Input Values
Input Layer
Hidden Layer 1
Hidden Layer 2
Output Layer

Model evaluation

- In deep learning involves assessing the performance of a trained neural network model on a set of test data. The goal of evaluation is to determine how well the model generalizes to new data that it has not seen during training.
- There are several metrics that can be used to evaluate the performance of a deep learning model, including:

1. Accuracy: the percentage of correctly classified samples in the test set.
2. Precision: the percentage of correctly classified positive samples out of all samples classified as positive.
3. Recall: the percentage of correctly classified positive samples out of all actual positive samples.
4. F1 score: a harmonic mean of precision and recall that balances between the two metrics.
5. Area Under the Curve (AUC): a measure of how well the model is able to distinguish between positive and negative samples, based on the true positive rate and false positive rate.

evaluations of the model

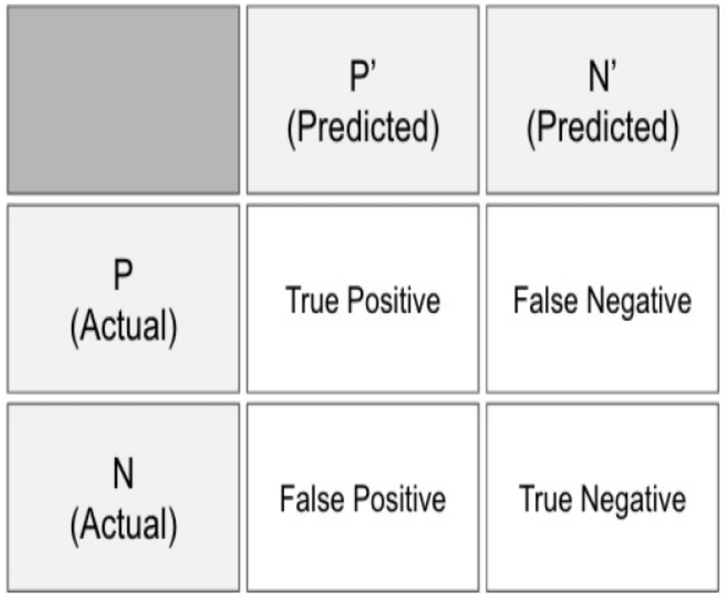


```
Accuracy:  0.94
Precision: 0.8662
Recall:    0.8955
F1 Score:  0.8806
```

- Four different common measures in the evaluation of machine learning models.

Evaluating Models

- the process of understanding how well they give the correct classification and then measuring the value of the prediction in a certain context
- We care how model gets any prediction correct
- **The Confusion Matrix** also called a *table of confusion*
- table of rows and columns that represents the predictions and the actual outcomes (labels) for a classifier
- True positives (Positive prediction - Label was positive)
- False positives (Positive prediction - Label was negative)
- True negatives (Negative prediction - Label was negative)
- False negatives (Negative prediction - Label was positive)



	P' (Predicted)	N' (Predicted)
P (Actual)	True Positive	False Negative
N (Actual)	False Positive	True Negative

Confusion matrix

- The confusion matrix is a table used to evaluate the performance of a classification model. It provides a detailed breakdown of how the model classified the samples in the test set, by comparing the predicted labels with the actual labels.
- The confusion matrix is typically represented as a table with rows and columns corresponding to the actual and predicted classes, respectively. Each cell in the table represents the number of samples that belong to the corresponding actual and predicted class.

The four terms in the matrix

- True Positive (TP): samples that are actually positive and were correctly classified as positive
- False Positive (FP): samples that are actually negative but were incorrectly classified as positive
- True Negative (TN): samples that are actually negative and were correctly classified as negative
- False Negative (FN): samples that are actually positive but were incorrectly classified as negative

Sensitivity versus specificity

- *Sensitivity* and *specificity* are two different measures of a binary classification model.
- The true positive rate measures how often we classify an input record as the positive class and its the correct classification.
- This also is called sensitivity or recall
- Like classifying a patient as having a condition who was actually sick. Sensitivity quantifies how well the model avoids false negatives.
- Sensitivity = TP / (TP + FN)
- Example as not having the condition and actually did not have the condition then this would be considered a true negative (also called specificity).
- Specificity = TN / (TN + FP)

Accuracy

- Accuracy is the degree of closeness of measurements of a quantity to that quantity’s true value.
- Accuracy = (TP + TN) / (TP + FP + FN + TN)
- Accuracy can be misleading in the quality of the model when the class imbalance is high.

Precision

- The degree to which repeated measurements under the same conditions give us the same results
- as the *positive prediction value*
- Precision = TP / (TP + FP)

Recall and F1

- as sensitivity and is also known as the *true positive rate*
- In binary classification we consider the F1 score (or F-score, F-measure) to be a measure of a model's accuracy
- The F1 score is the harmonic mean of both the precision and recall measures
- F1 = 2TP / (2TP + FP + FN)
- F1 between 0.0 and 1.0, where 0.0 is the worst score and 1.0 is the best score
- F1 score used as an overall score on how well our model is performing.
- Score = MIN(Precision, Recall)

Receiver Operating Characteristic (ROC)

- The Receiver Operating Characteristic (ROC) curve is a graphical representation of the performance of a binary classification model as the discrimination threshold is varied.
- The ROC curve plots the true positive rate (TPR) on the y-axis and the false positive rate (FPR) on the x-axis. The TPR is the proportion of actual positive samples that are correctly classified as positive, while the FPR is the proportion of actual negative samples that are incorrectly classified as positive.
- To create the ROC curve, the classification model's output is sorted by predicted probability or score, and a threshold is set to decide the class label. As the threshold is varied, the TPR and FPR are calculated at each threshold value, and these values are used to plot the ROC curve.

ROC reading

- A perfect classifier would have a TPR of 1 and an FPR of 0, resulting in a point at the top left corner of the plot. A random classifier would have a diagonal line from the bottom left corner to the top right corner, while a classifier that performs worse than random would have a curve that falls below the diagonal.
- The area under the ROC curve (AUC) is a commonly used metric to evaluate the performance of a binary classification model. The AUC measures the overall ability of the model to discriminate between positive and negative samples, regardless of the specific threshold used. A model with an AUC of 1.0 is a perfect classifier, while a model with an AUC of 0.5 is no better than random guessing.

ROC

An **ROC curve** (**receiver operating characteristic curve**) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

- True Positive Rate
- False Positive Rate

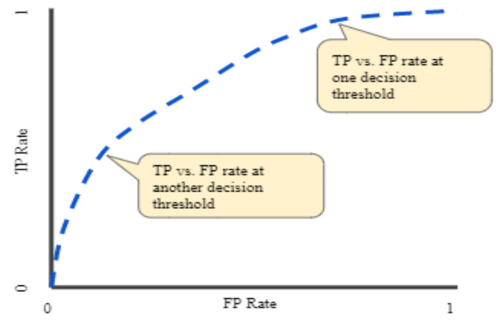
True Positive Rate (**TPR**) is a synonym for recall and is therefore defined as follows:

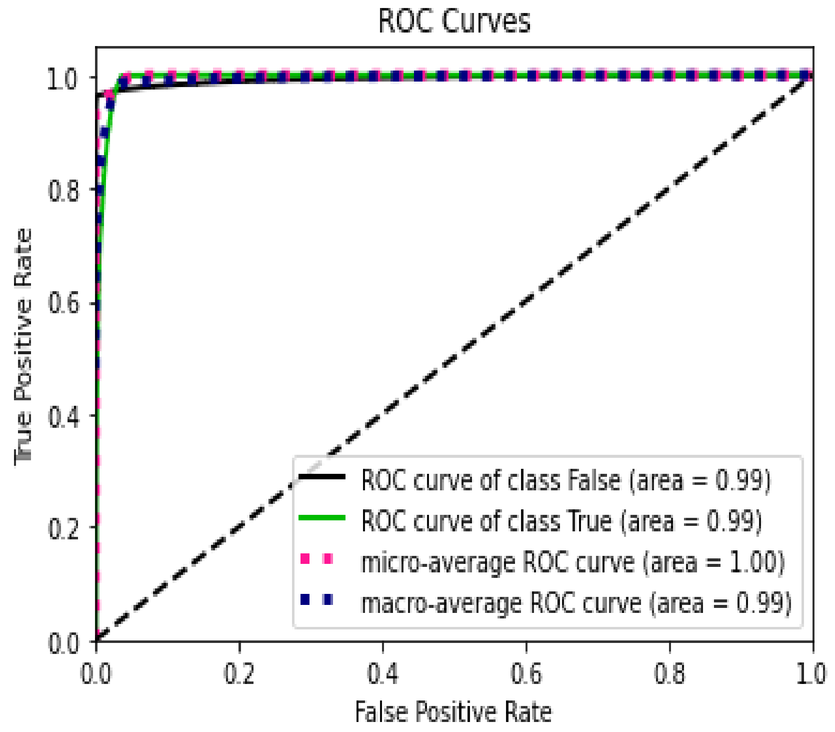
$$TPR = \frac{TP}{TP + FN}$$

False Positive Rate (**FPR**) is defined as follows:

$$FPR = \frac{FP}{FP + TN}$$

An ROC curve plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. The following figure shows a typical ROC curve.





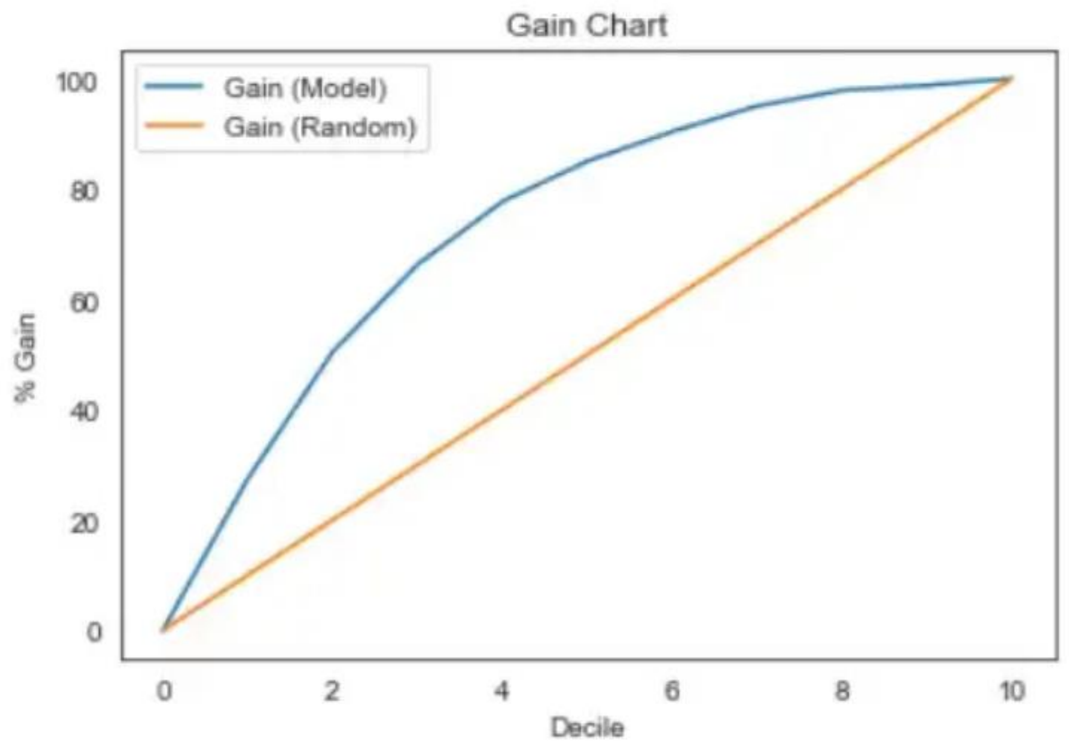
Lift and gain

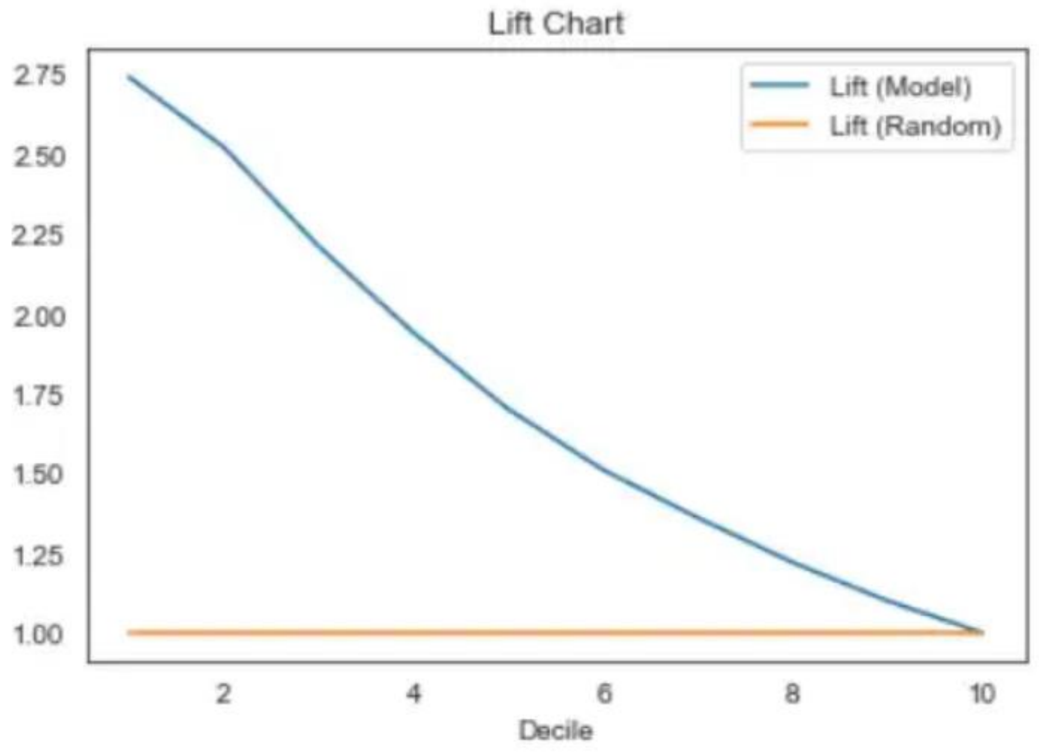
- Lift and gain are two evaluation metrics commonly used in marketing and sales to assess the effectiveness of targeted marketing campaigns.
- Lift measures how much more likely the target audience is to respond to a marketing campaign compared to the general population. It is calculated as the ratio of the response rate of the target audience to the response rate of the general population, after adjusting for the size of the target audience. A lift of 1 indicates that the campaign had no effect, while a lift greater than 1 indicates that the campaign was effective in increasing the response rate.
- Gain, on the other hand, measures the absolute increase in response rate due to the campaign, compared to the response rate without the campaign. It is calculated as the difference between the response rate of the target audience with the campaign and the response rate of the target audience without the campaign

Lift and gain

- Accuracy, Precision, F1 Score, ROC-AUC
- Sometimes these metrics did not reflect how the model would do business-wise
- how our model's differences would make in the business term compared to random targeting.
- use Lift and Gain analysis **To measure how much better our prediction model compared without the model**.
- Lift and Gain analysis is an analysis to evaluate the model prediction and the benefit to the business.
- **Lift** is a measure of the effectiveness of a predictive model calculated as the ratio between the results obtained with and without the predictive model.
- The analysis result would be presented in the chart
- **Gain** is the ratio between the cumulative number of the Number of Responses (Actual Positive) up to each decile divided by the total number of positive observations in the data

Lift and Gain charts





Collinearity

- In statistics, **collinearity** refers to a linear relationship between two explanatory variables. Two variables are perfectly collinear if there is an exact linear relationship between the two, so the correlation between them is equal to 1 or −1.
- A **collinearity** is a special case when two or more variables are exactly correlated.
- Reduces the precision of the estimate coefficients, which weakens the statistical power of your regression **model**. You might not be able to trust the p-values to identify independent variables that are statistically significant.
- Principal Component Analysis or Autoencoders
- Luckily, **decision trees** and boosted **trees** algorithms are immune to **multicollinearity** by nature and DNNs are usually fine
- Can make models more difficult to interpret
- Collinearity will inflate the variance and standard error of coefficient estimates

Collinearity

- Collinearity, also known as multicollinearity, is a phenomenon that occurs when two or more predictor variables in a regression model are highly correlated with each other. This can lead to several issues in regression analysis, such as unstable or unreliable estimates of the regression coefficients, difficulty in interpreting the individual effects of the predictors on the response variable, inflated standard errors of the estimates.
- Collinearity can arise when two or more predictor variables are measuring similar or redundant information, or when they are on different scales or units of measurement. One way to detect collinearity is to calculate the correlation matrix between the predictor variables and look for high correlation coefficients (e.g., greater than 0.7 or 0.8).

Detecting Collinearity

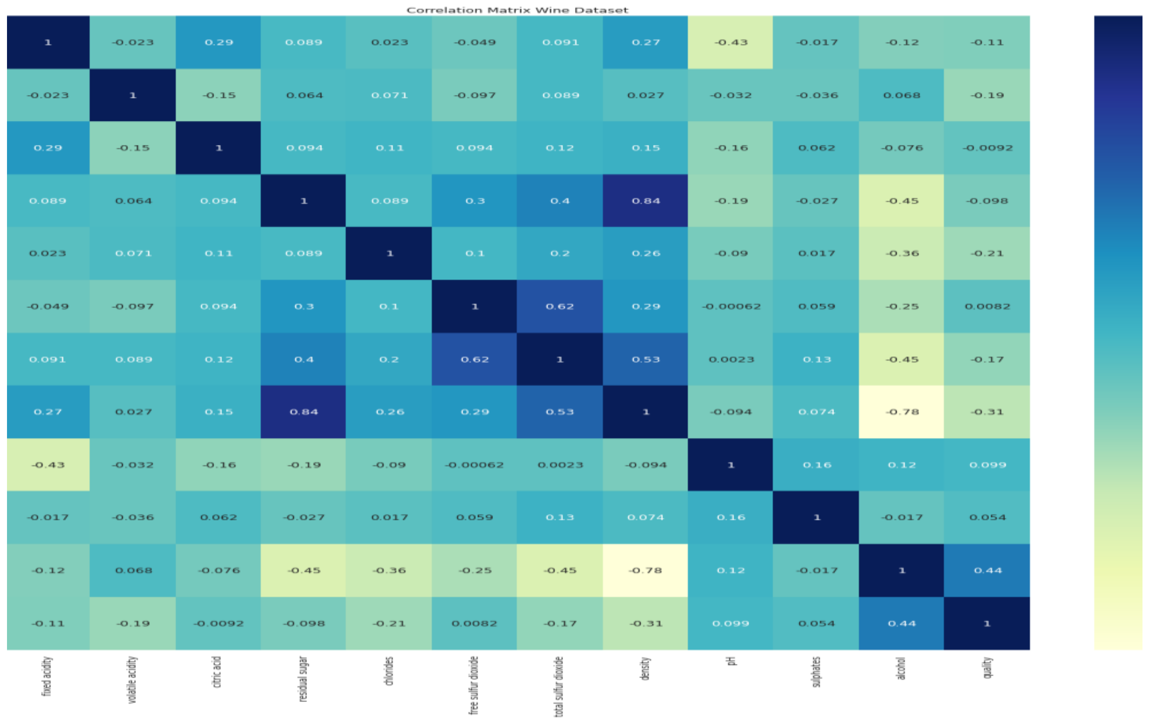
- There are two easy ways to detect if collinearity exists in our regression model.
- **Correlation Matrix**
  - A correlation matrix is simply **a table which displays the correlation coefficients for different variables**
- **Variance Inflation Factor**
  - the influence of collinearity on the variance of our coefficient estimates

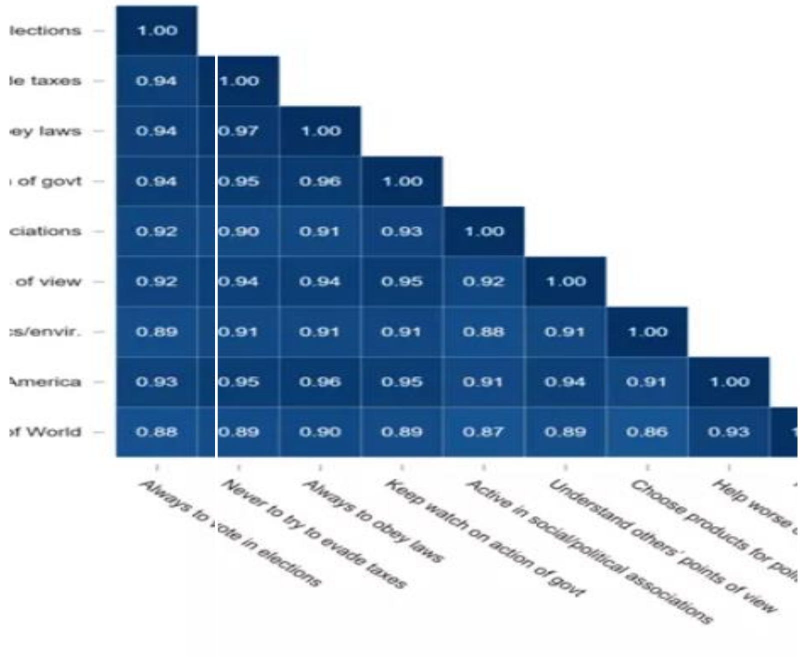
$$VIF = \frac{1}{1-R_i^2}$$

A correlation matrix

- A correlation matrix is a table that displays the correlation coefficients between pairs of variables in a dataset. It is a useful tool for exploring the relationships between multiple variables and identifying potential patterns or dependencies among them.
- The correlation coefficient is a statistical measure that ranges between -1 and +1, indicating the strength and direction of the linear relationship between two variables. A value of +1 indicates a perfect positive correlation, where the two variables increase or decrease together in a linear fashion. A value of -1 indicates a perfect negative correlation, where the two variables move in opposite directions. A value of 0 indicates no correlation between the two variables.
- In a correlation matrix, the diagonal entries are always 1, as each variable is perfectly correlated with itself. The upper and lower triangles of the matrix display the correlation coefficients between pairs of variables, with the upper triangle showing the same values as the lower triangle since the correlation between variable A and variable B is the same as the correlation between variable B and variable A

Correlation Matrix

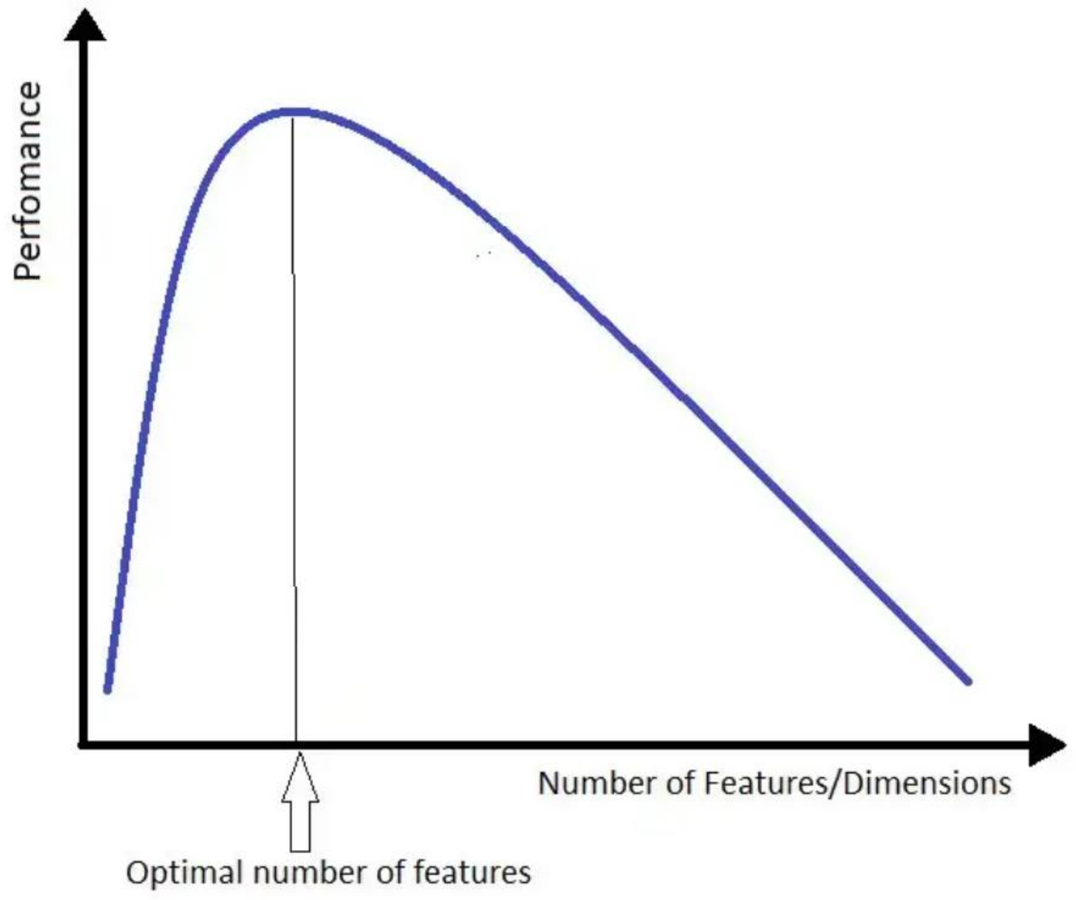




- https://www.displayr.com/what-is-a-correlation-matrix/

Curse of Dimensionality

- Curse of Dimensionality describes the explosive nature of increasing data dimensions and its resulting exponential increase in computational efforts required for its processing and/or analysis.
- to explain the increase in volume of Euclidean space associated with adding extra dimensions, in area of dynamic programming.
- As the dimensionality increases, the number of data points required for good performance of any machine learning algorithm increases exponentially.
- Simpler Version: Too many features can create noise and based on amount of training data, can cause the model to not predict correctly
- Feature Engineering to do Feature Reduction is essential



Regularization

- One of the major aspects of training your machine learning model is avoiding overfitting
- *The model will have a low accuracy if it is overfitting*
- model is trying too hard to capture the noise in your training dataset.
- *noise we mean the data points that don't really represent the true properties of your data, but random chance*
- Regularization is a set of techniques that can prevent overfitting in neural networks and thus improve the accuracy of a Deep Learning model when facing completely new data from the problem domain
- *Regularization, significantly reduces the variance of the model, without substantial increase in its bias*

```
from keras import regularizers
model.add(Dense(64, input_dim=64,   kernel_regularizer=regularizers.l2(0.01)
```

Regularization

- Regularization is a technique used in machine learning to prevent overfitting and improve the generalization performance of models. Overfitting occurs when a model is too complex and fits the training data too closely, resulting in poor performance on new, unseen data. Regularization methods add a penalty term to the model's cost function that encourages the model to have simpler or smoother parameter values, reducing the complexity of the model.
- There are several types of regularization techniques, including L1 regularization, L2 regularization, and elastic net regularization.

types of regularization techniques

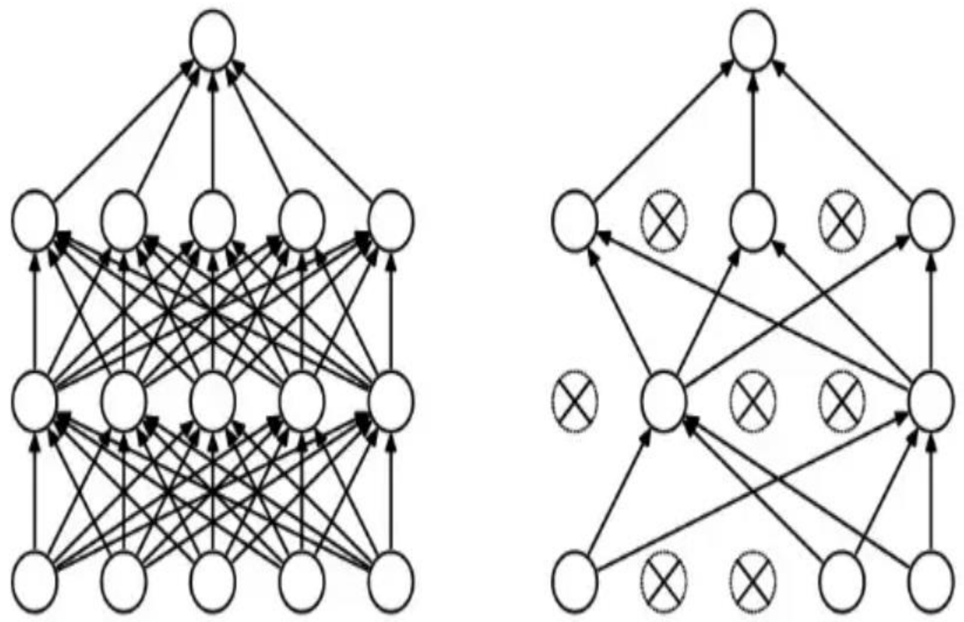
- L1 regularization, also known as Lasso regularization, adds a penalty term to the cost function proportional to the absolute value of the model's parameter values. This has the effect of shrinking some parameter values to zero, resulting in a sparse model that selects only the most relevant features for the prediction task.
- L2 regularization, also known as Ridge regularization, adds a penalty term proportional to the squared value of the model's parameter values. This has the effect of shrinking all parameter values towards zero, resulting in a model that is more stable and less sensitive to small changes in the input data.
- Elastic net regularization combines both L1 and L2 regularization, resulting in a model that has both sparse and stable parameter values.

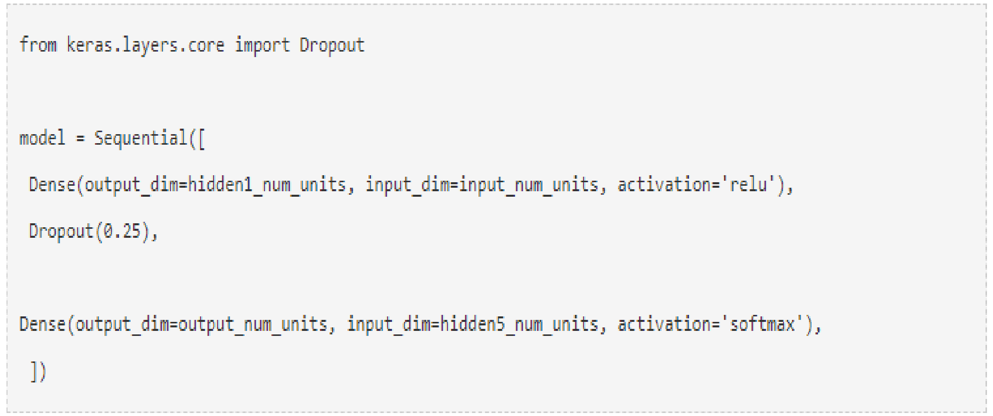
Dropout

- Dropout is a regularization technique used in neural networks to prevent overfitting and improve the generalization performance of models. It works by randomly dropping out (i.e., setting to zero) a percentage of the neurons in the network during training.
- During each training iteration, a certain percentage of neurons in each layer of the network are randomly selected to be dropped out. This means that each neuron is only activated with a certain probability, and the network learns to make predictions using only a subset of the available neurons. This has the effect of making the network more robust and preventing overfitting.
- Dropout has several advantages over other regularization techniques. It is simple to implement, computationally efficient, and can be applied to any type of neural network. Additionally, dropout has been shown to improve the generalization performance of models in various tasks, such as image classification, natural language processing, and speech recognition.

Dropout

- *The term "dropout" refers to dropping out the nodes (input and hidden layer) in a neural network*
- Randomly select some nodes and removes them along with all their incoming and outgoing connections

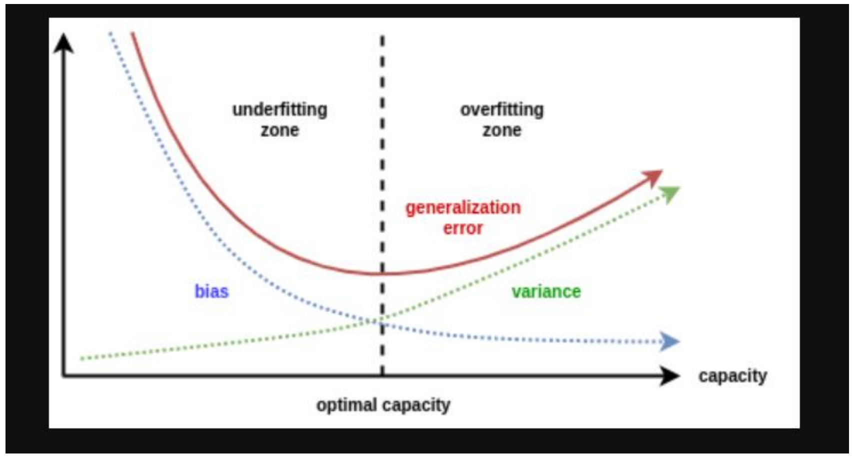

(a) Standard Neural Net (b) After applying dropout.


```
from keras.layers.core import Dropout


model = Sequential([
 Dense(output_dim=hidden1_num_units, input_dim=input_num_units, activation='relu'),
 Dropout(0.25),


Dense(output_dim=output_num_units, input_dim=hidden5_num_units, activation='softmax'),
 ])
```

https://towardsdatascience.com/dropout-in-neural-networks-47a162d621d9

Early Stopping

- Early Stopping is a regularization technique for deep neural networks that stops training when parameter updates no longer begin to yield improves on a validation set.
- *Early Stopping, does not only protect against overfitting but needs considerably less number of Epoch to train.*



```
from keras.callbacks import EarlyStopping


EarlyStopping(monitor='val_err', patience=5)
```

Batch Normalization

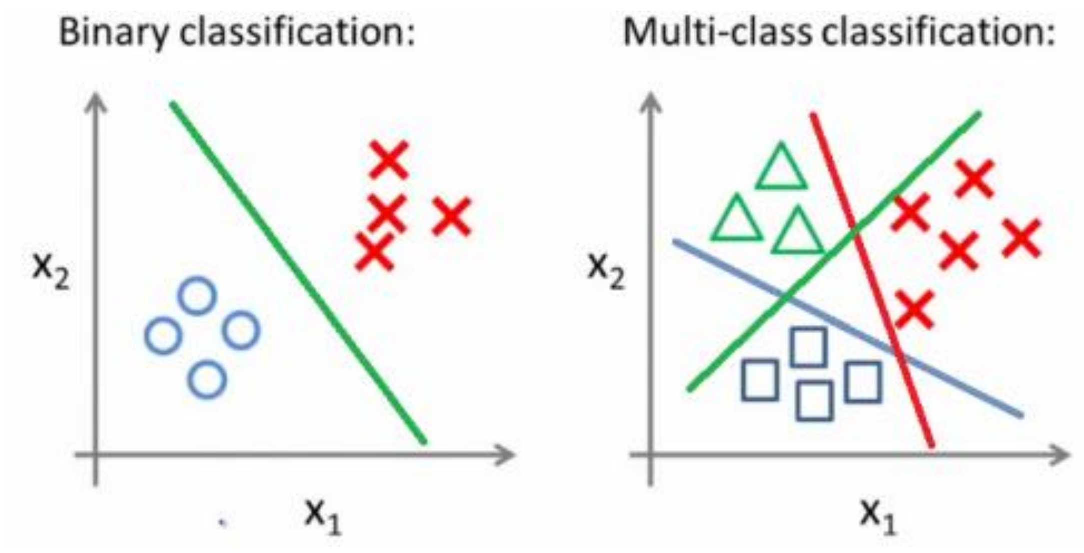
- ***Batch normalization layers*** act similar to the data preprocessing steps mentioned earlier
  - They calculate the mean μ and variance σ of a batch of input data, and normalize the data *x* to a zero mean and unit variance
  - I.e., $\hat{x} = \frac{x-\mu}{\sigma}$
- BatchNorm layers alleviate the problems of proper initialization of the parameters and hyper-parameters
  - Result in faster convergence training, allow larger learning rates
  - Reduce the internal covariate shift
- BatchNorm layers are inserted immediately after convolutional layers or fully-connected layers, and before activation layers
  - They are very common with convolutional NNs

Batch normalization

- Batch normalization is a technique used in deep neural networks to improve training stability and performance by normalizing the inputs of each layer.
- The idea behind batch normalization is to normalize the inputs of each layer by subtracting the mean and dividing by the standard deviation of the inputs. This has the effect of standardizing the inputs and making them more suitable for training.
- Batch normalization is typically applied after the activation function of each layer, and before the next layer's weights are applied. During training, batch normalization is performed over small batches of training examples, rather than the entire training set, which allows for greater computational efficiency.
- Batch normalization has several benefits. First, it can improve training stability by reducing the internal covariate shift, which is the phenomenon where the distribution of layer inputs changes during training, making it harder for the network to learn. By normalizing the inputs of each layer, batch normalization reduces the internal covariate shift and stabilizes training.

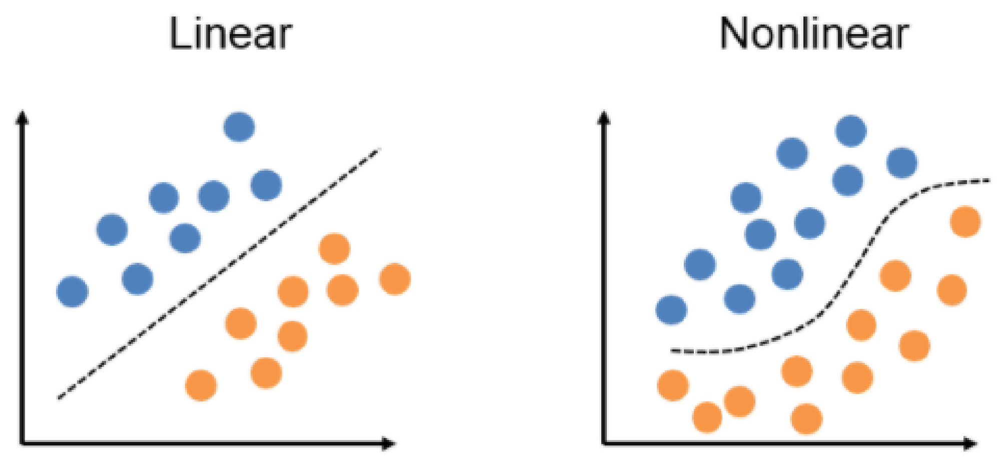
Binary vs Multi-class Classification

- A classification problem with only 2 classes is referred to as ***binary classification***
    - The output labels are 0 or 1
    - E.g., benign or malignant tumor, spam or no-spam email
- A problem with 3 or more classes is referred to as ***multi-class classification***



Binary vs Multi-class Classification

- Both the binary and multi-class classification problems can be linearly or non-linearly separated
  - Figure: linearly and non-linearly separated data for binary classification problem



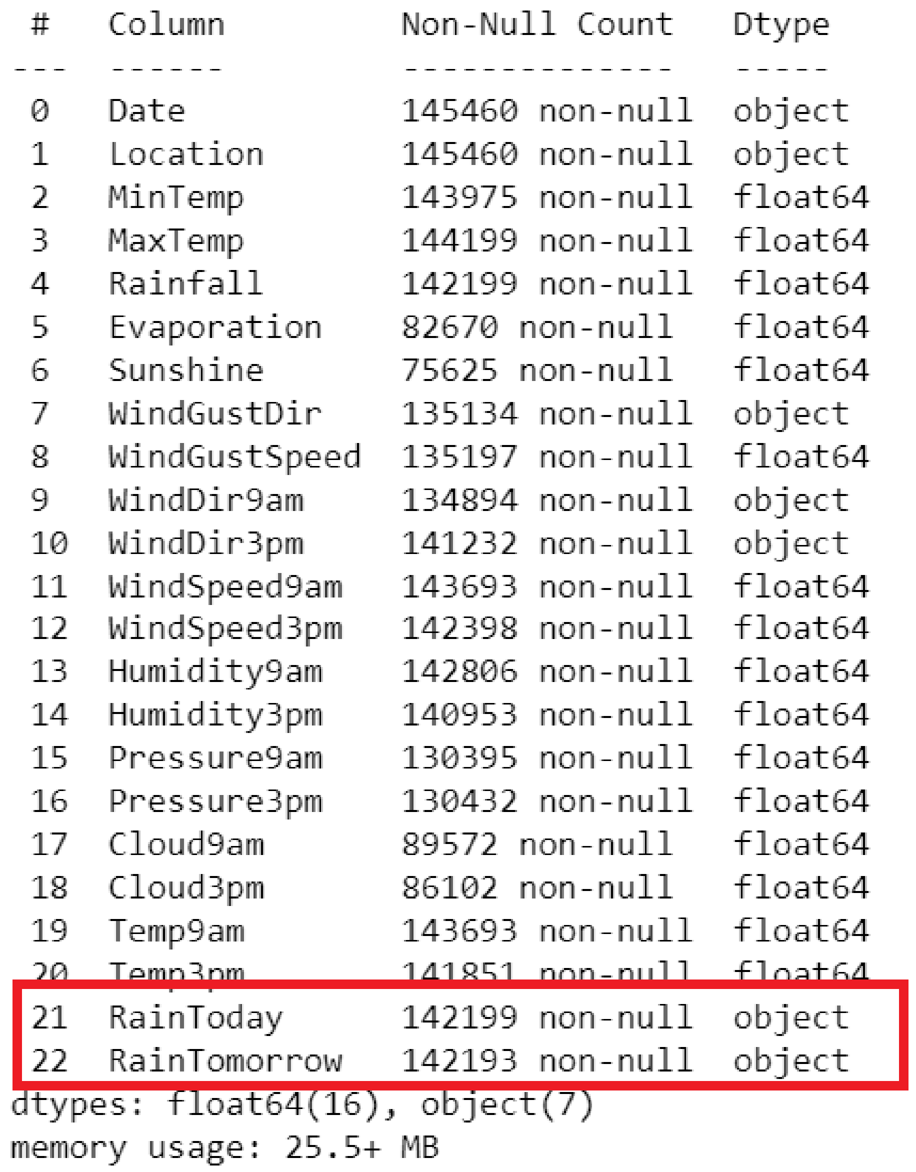
Coding time

Importing libraries

- %matplotlib inline
- import seaborn as sns
- import matplotlib.pyplot as plt
- import pandas as pd
- import numpy as np
- import datetime
- data = pd.read_csv('C:/weatherAUS.csv')
- data.head()

Data Exploration

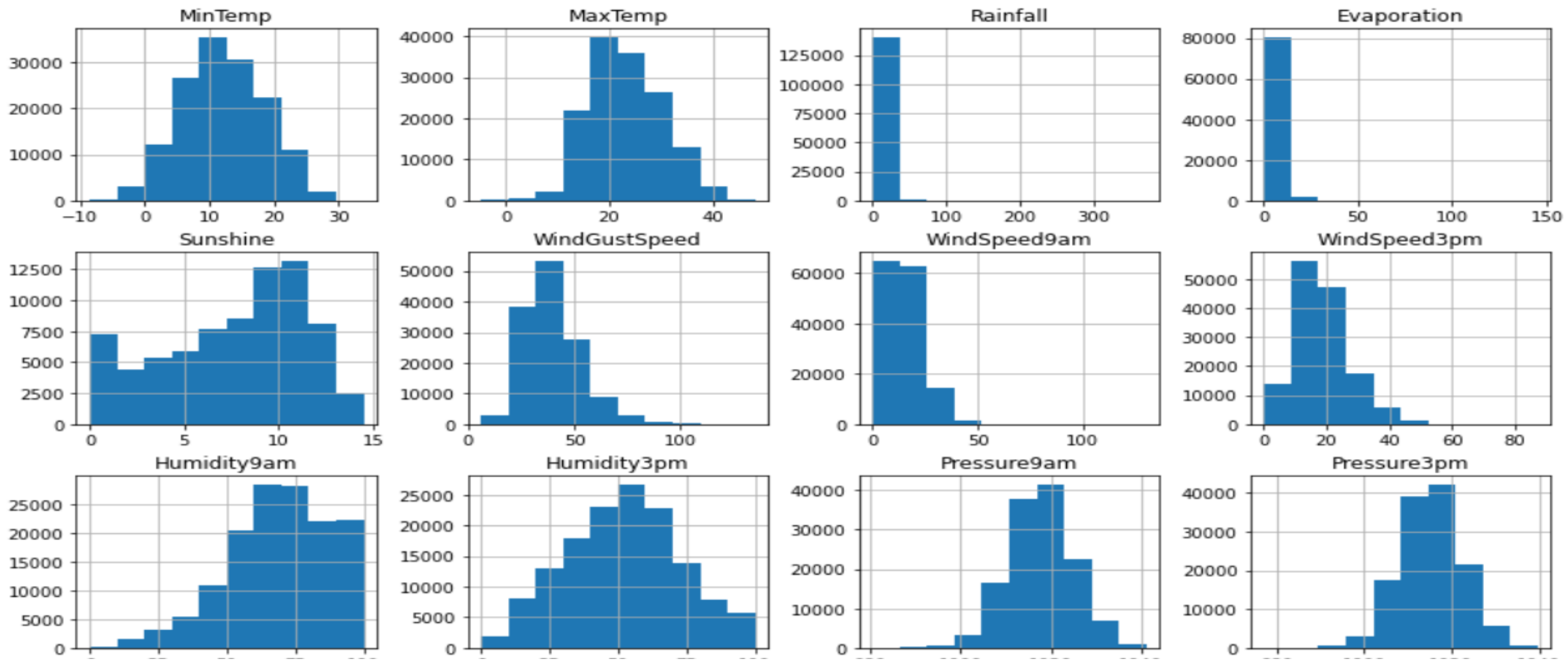
- data.head()
- data.shape
- data.info()
- data.isnull().sum()
- df.describe()



```
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Date           145460 non-null  object
 1   Location       145460 non-null  object
 2   MinTemp        143975 non-null  float64
 3   MaxTemp        144199 non-null  float64
 4   Rainfall       142199 non-null  float64
 5   Evaporation    82670 non-null   float64
 6   Sunshine       75625 non-null   float64
 7   WindGustDir    135134 non-null  object
 8   WindGustSpeed  135197 non-null  float64
 9   WindDir9am     134894 non-null  object
 10  WindDir3pm     141232 non-null  object
 11  WindSpeed9am   143693 non-null  float64
 12  WindSpeed3pm   142398 non-null  float64
 13  Humidity9am    142806 non-null  float64
 14  Humidity3pm    140953 non-null  float64
 15  Pressure9am    130395 non-null  float64
 16  Pressure3pm    130432 non-null  float64
 17  Cloud9am       89572 non-null   float64
 18  Cloud3pm       86102 non-null   float64
 19  Temp9am        143693 non-null  float64
 20  Temp3pm        141851 non-null  float64
 21  RainToday      142199 non-null  object
 22  RainTomorrow   142193 non-null  object
dtypes: float64(16), object(7)
memory usage: 25.5+ MB
```

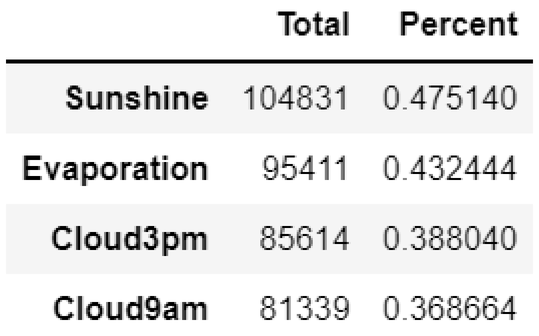
Data Exploring

- data.hist(figsize=(15,15))
- data.columns.values



Missing data

- total = oversampled.isnull().sum().sort_values(ascending=False)
- percent = (oversampled.isnull().sum()/oversampled.isnull().count()).sort_values (ascending=False)
- missing = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
- missing.head(4)



	Total	Percent
Sunshine	104831	0.475140
Evaporation	95411	0.432444
Cloud3pm	85614	0.388040
Cloud9am	81339	0.368664

Drop null tables

- df = pd.get_dummies(data, columns=['Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm'])
- df = df.drop(columns=['Date'])
- df.describe()

- df.columns.values

Thank You