45.1 Introduction

Featurization/Feature Engineering is the most important aspect of Machine Learning. Featurization is about converting the data of one type into a numerical vector form. Feature Engineering is about modifying the features so that they make the machine learning models perform well at prediction.

Out of the various featurization techniques available for the text data, the most commonly used techniques are

a) Bag of Words (BOW)
b) Term Frequency - Inverse Document Frequency (TF-IDF)
c) Average Word2Vec
d) TF-IDF Weighted Average Word2Vec

For the categorical features, the most commonly used featurization techniques are the mean response time and one-hot encoding.

Feature Engineering is the process of using domain knowledge to extract the new variables from the raw data, which makes the machine learning algorithms work better. Feature Engineering is a difficult task as it is domain-specific.

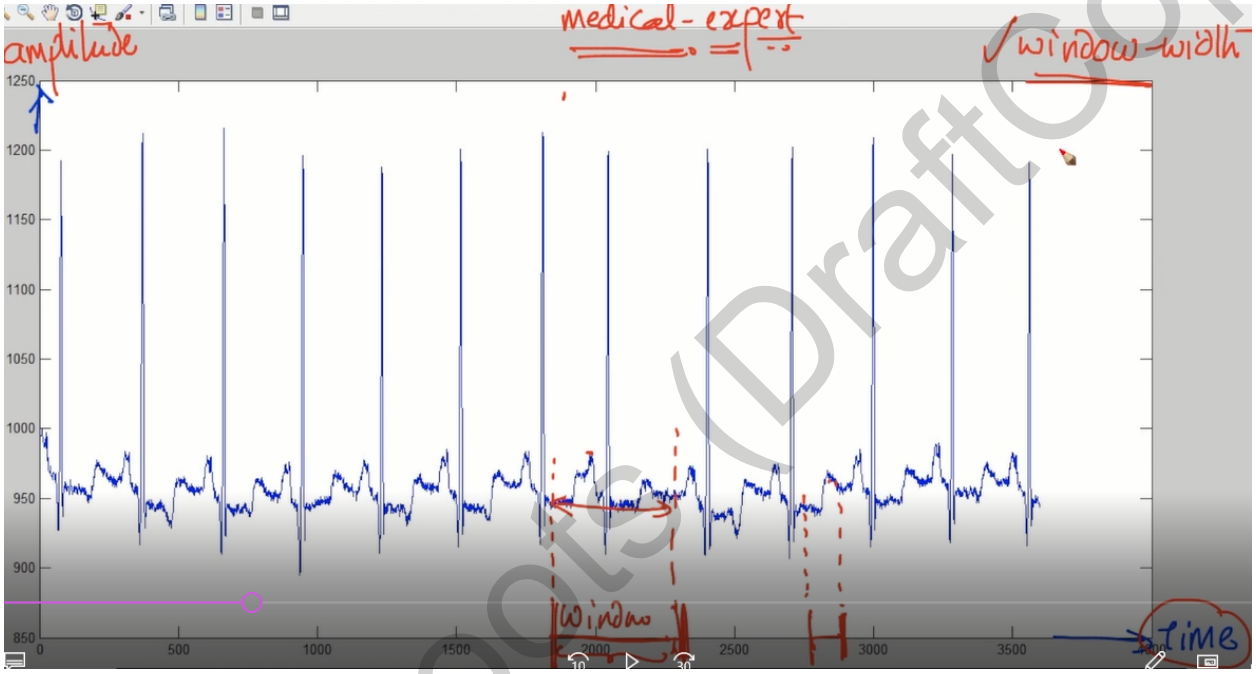
Feature Selection is to select some useful, relevant features among the features we generate/find/have in our data.

Note

If we are using a linear model like Logistic Regression or Linear SVM, then we perform feature engineering to convert the non-linearly separable data into a linearly separable format, to make our models work better. In a nutshell, we are trying to transform our features to better suit our models and hence design better overall models.

45.2 Moving Window for Time Series Data

Moving window is the simplest featurization technique applied to the time series data. Let us take the example of ECG/Medical data, which measures the heartbeat. We have to convert this time series data into numerical vector form, and for that, we have to sit with the medical/domain expert.



Some of the standard featurization techniques that are performed on the data, that is picked from this window are

a) Mean, Standard Deviation
b) Median, Quantiles
c) Max, Min, Max-Min, Max/Min (or) local minima/maxima
d) Zero-Crossing (checking how many times, the signal has crossed zero).

All these featurization techniques are domain-specific. Before going with any featurization technique, we should seek expert advice.

Example

Let us consider the problem of predicting cardiac arrest in the next 10 minutes.

Steps of Featurization

1) Define Window Width
2) Find out the features in this window that are useful in predicting heart-attack.
3) After getting the unimportant features, take the combination of those features as 'x_i' and the response binary variable as 'y_i'. (Here the response variable indicates cardiac arrest occurs or not)

For any problem, we have to do some domain-based research in order to decide which features are important for the task and which featurization technique has to be applied.

Q) Why is this approach called the Moving Window approach?

Ans) This approach is called the Moving Window approach because a window of defined size and shape, is moved over the data, the moving distance is equal to the width of the window.

Q) How to determine the width of the window? Is it mandatory for all the windows to be of the same size?

Ans) In case if we are taking multiple windows into consideration, we need to take all the windows of the same width. But we might get overlapping windows where one window overlaps over another in case of having multiple windows of different widths.

Choosing windows of different widths would lead to some implementation challenges like

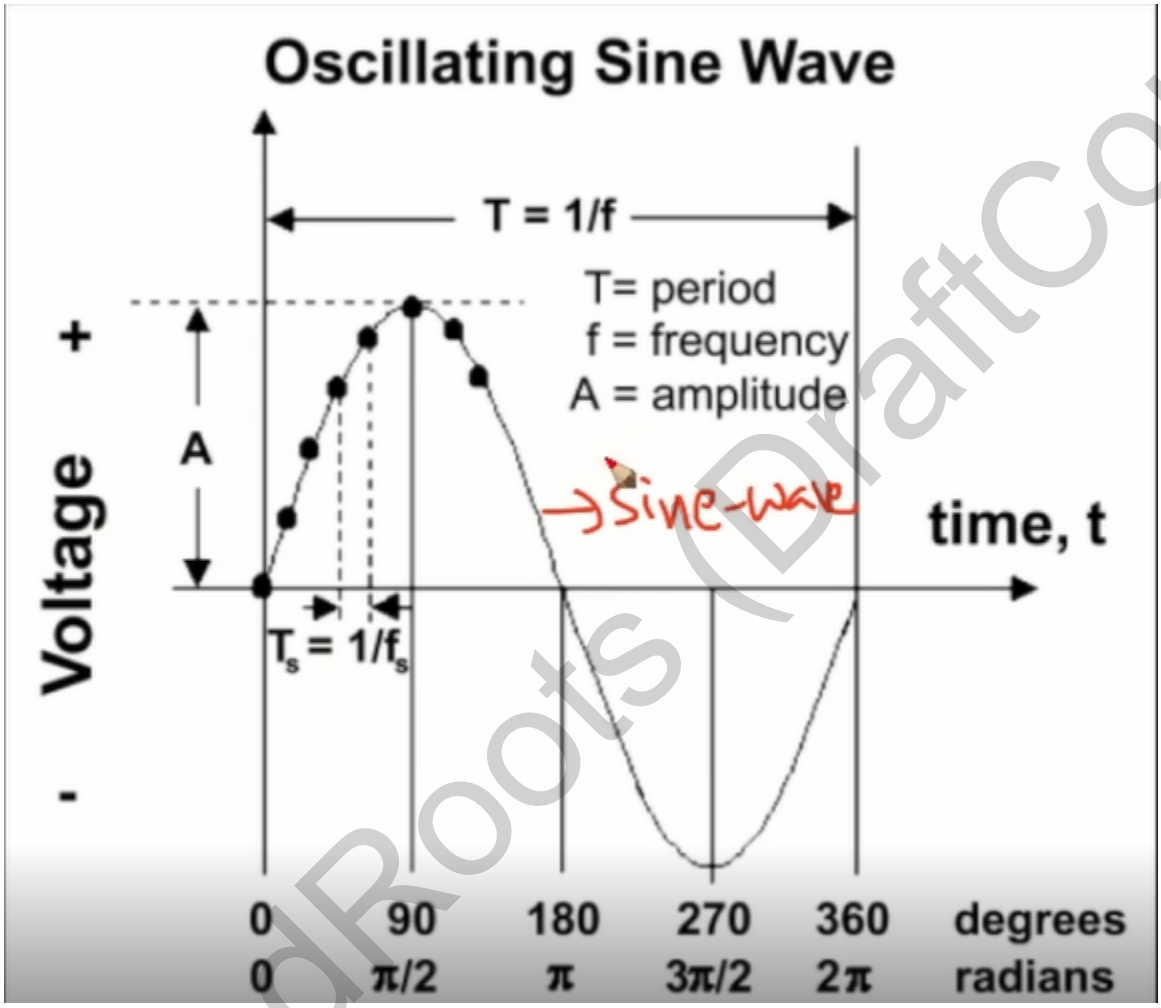
(i) How to determine the width of each window?

(ii) If we have just one width for all the windows, we can treat it as a hyperparameter. If we have multiple windows, the number of hyperparameters would increase and it would be very hard to tune all these hyperparameters.

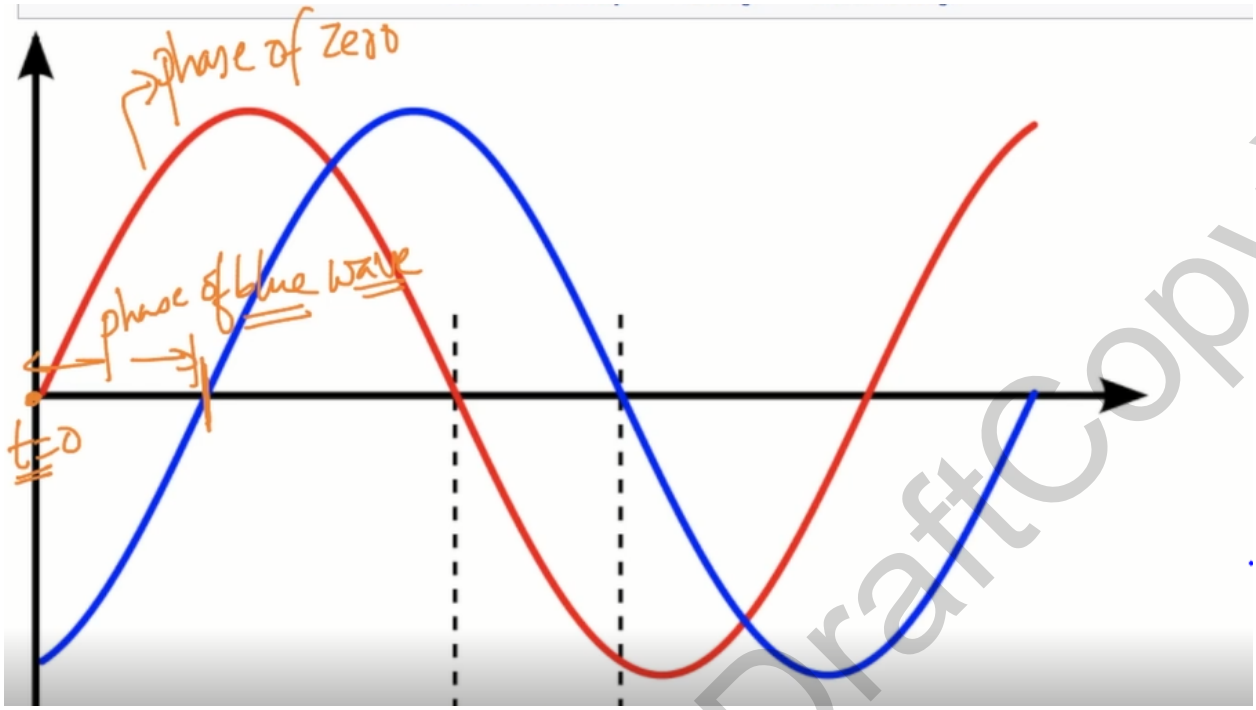
In practice, there is going to be no benefit in having a variable-sized window.

45.3 Fourier Decomposition

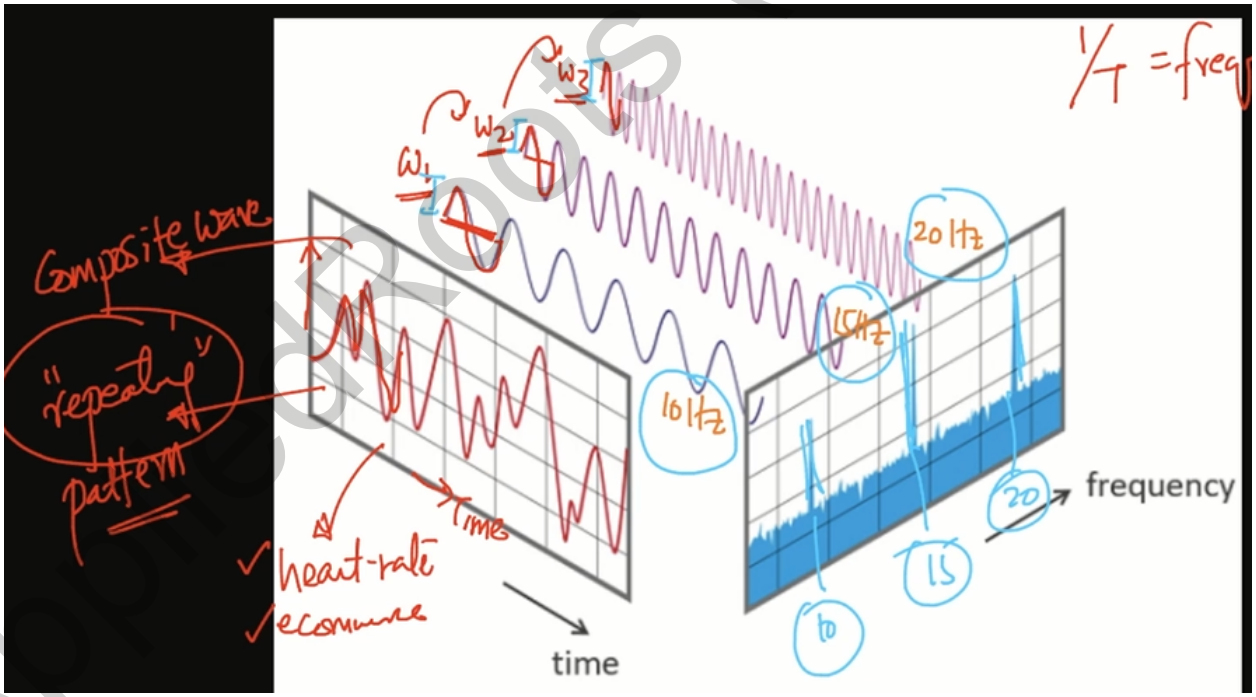
One of the most popular ways to represent the time series data is using the Fourier transform.



The number of oscillations per second is called **Frequency**.
The time taken to complete one cycle is called the **Time period**.
The height of a signal from '0' is called **Amplitude**.



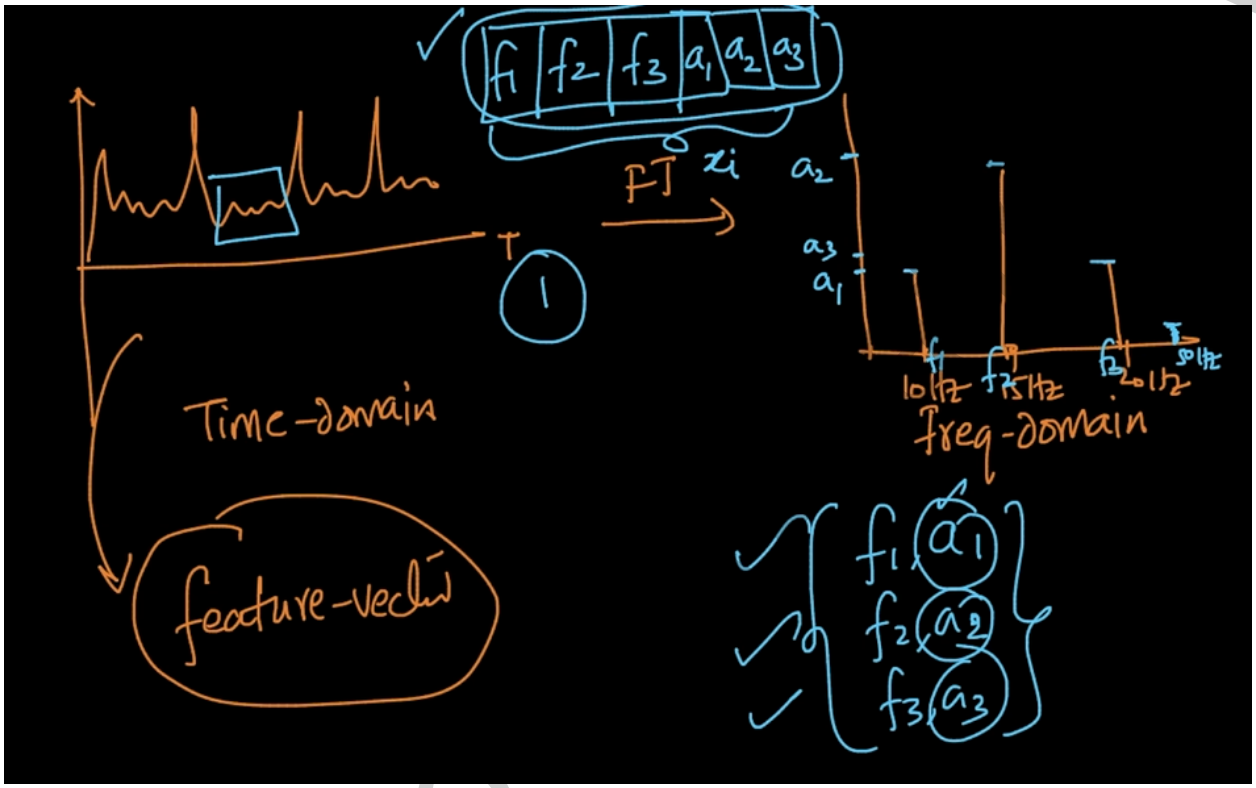
If the given data is in the form of a composite function/wave, we can break it into a sum of multiple sine waves. It applies to repeating patterns.



So after obtaining the frequency domain representation of the input time signal, we then have to obtain the numerical vector in the form

f_1	f_2	f_3	a_1	a_2	a_3

This is the **Fourier representation** (or) **Fourier representation** of the time series data. Fourier representation is always preferred when the given time series data has repeating waveforms. The Fourier transform breaks the given composite signal into multiple waves (each wave has a different time period and frequency)



The representation of the signal in Frequency domain by applying the Fourier Transform is necessary if there are repeating patterns in the given input Time Series signal. In such cases, the frequencies are very important.

45.4 Deep Learning Features: LSTM

Deep Learning models take lots of data as input and learn the best featurization techniques for the given data automatically. These features developed using the deep learning models are called Deep Learnt Features. These features are almost the best features.

Deep Learning models give the best results and features when the input data is in time series, text, or image formats. Deep Learning models should be used only when we have lots of data. It is not recommended to use the Deep Learning models if we have a small amount of input data.

Q) Why don't we use deep learning models instead of other models, when we have lots of data?

Ans) When we have lots of data, the deep learning models tend to work better than any other classical ML model. When we have the data in the form of images, videos, audio, or if we are working on complex tasks like machine translation of the text from one language to another, etc, the deep learning models work better. Based on the features we have, simple models like Logistic Regression, Support Vector Machine, Decision Tree, Random Forest, etc work fairly well.

If the model interpretability is important in the problem, then using the classic ML algorithms is better, instead of using the deep learning models.

If we are working on low latency applications, where if a query point 'x_q' is given, the time taken to give the response 'y_q' should be of the order of milliseconds, then as the evaluation time of the deep learning models is high, it is better to go with classical ML algorithms.

Q) How are the deep learning models different from the classical ML models?

Ans) In a deep learning model, we featurize the data using all of the initial layers and the final classification is done by the last layer which is often a Logistic Regression (or) an extension to it (or) a similar method. So the deep learning models perform feature engineering and classification (or) regression in a single model.

While in classical ML algorithms, we featurize the data using techniques like Fourier Transforms and then input the transformed features to the model, and then classification (or) regression is performed.

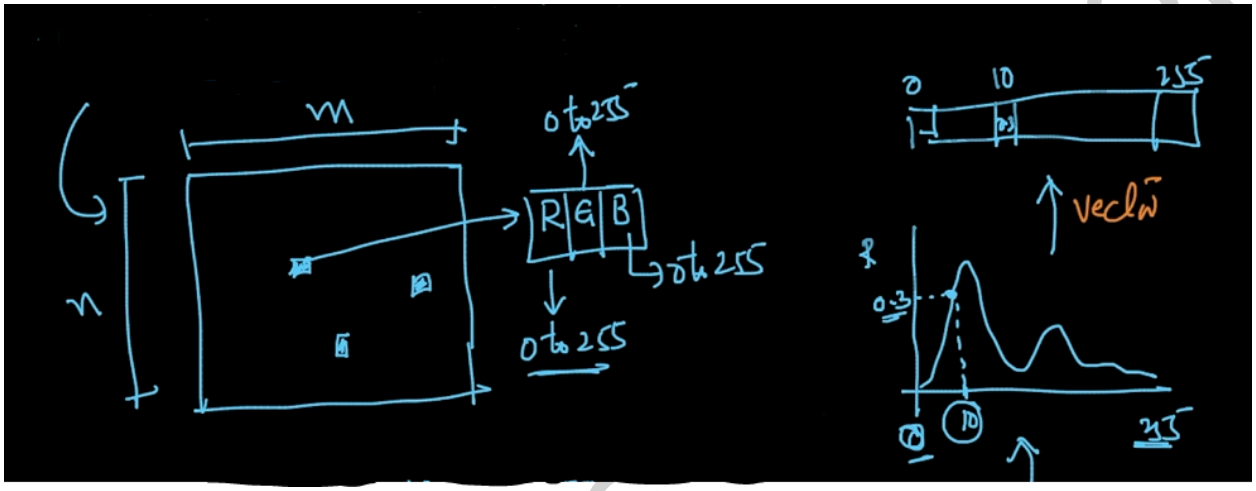
The decision of choosing a deep learning model or a classical ML model depends on the real-world problem we are solving, the problem constraints, and the requirements of the problem to be fulfilled.

45.5 Image Histogram

There are two types of Image Histograms. They are

a) Color Histograms
b) Edge Histograms

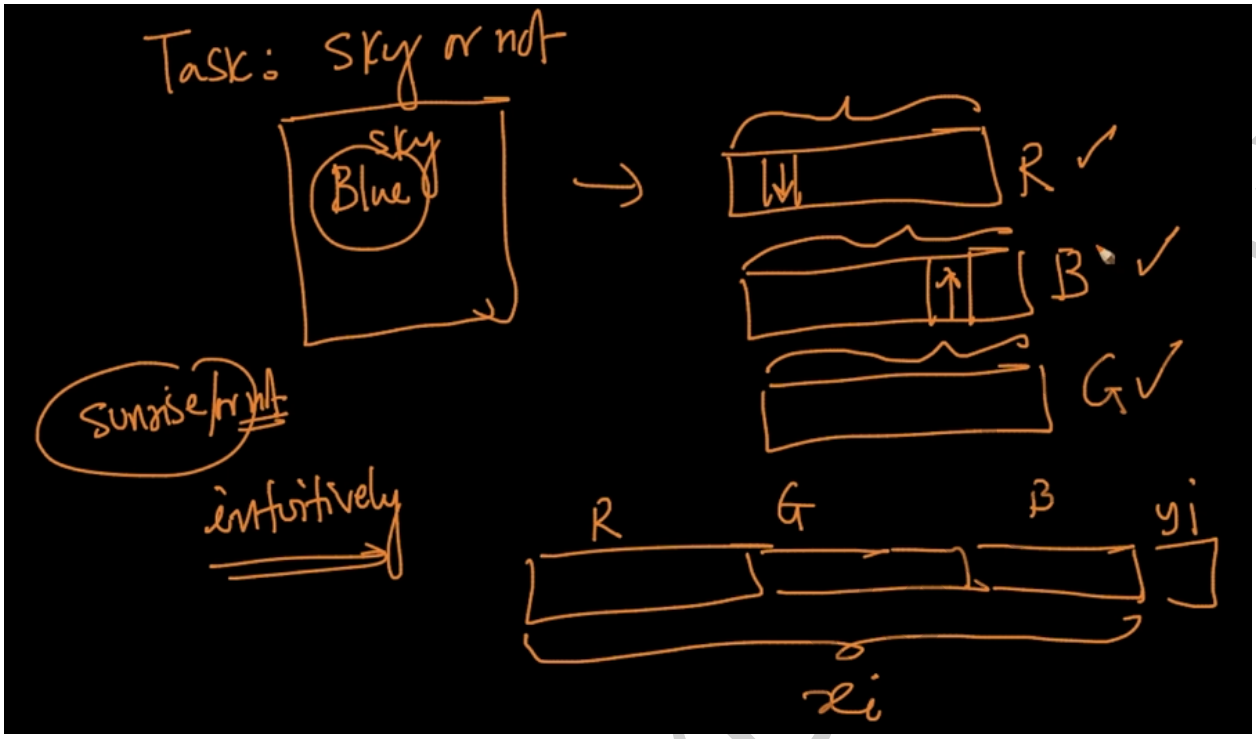
a) Color Histograms



Every image consists of pixels and every pixel consists of 3 values (R,G,B). There are a total of n*m pixels in the image. (where **nXm** is the dimensionality of an image)

We have to take the first component values(ie., values associated with red color) of all the pixels in the image, and have to plot the histograms with all possible values(ie., from 0 to 255). The possible values(ie., 0 to 255) are on the 'X' axis and their corresponding frequencies are on the 'Y' axis, and this histogram will be converted into a 256-dimensional vector, with the frequencies as the vector components.

Similarly, we have to convert the histograms of the green and the blue colors into 256-dimensional vectors each. For every color image, we get three 256-dimensional vectors. We then have to combine these three 256-dimensional vectors(ie., concatenation) and use the obtained 768-dimensional vector (3*256=768) for data modeling.

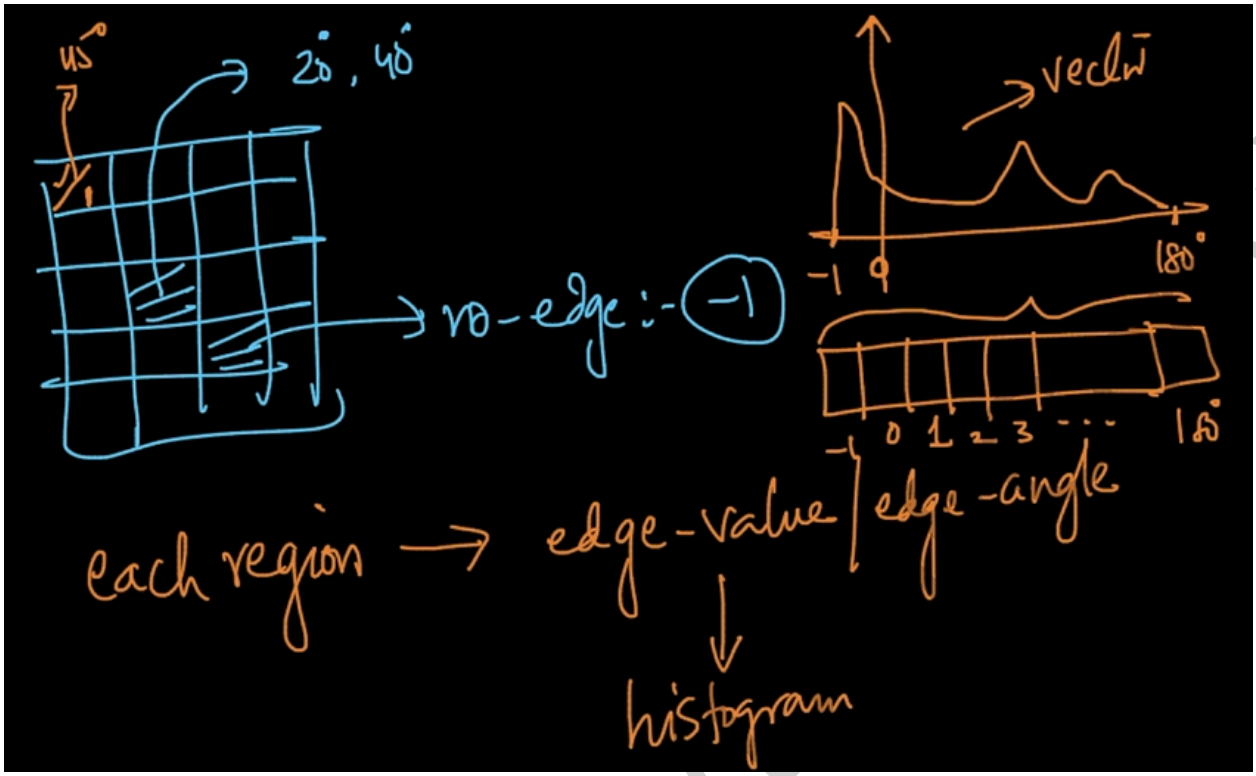


The color histogram can recognize only the colors, but not the shapes like square, rectangle, circle, etc. In order to recognize the shapes, we have another type of histogram called **Edge Histogram**.

b) Edge Histograms

Here we have to convert the image into a grid and in each cell, we have to check for an edge. If there is no edge in a cell, then the value in that cell becomes -1. In case, if we have multiple edges in a cell, then the edge with a larger angle is assigned to this cell.

So we get the edge angle/value to each of the cells in the grid. Now we have to plot a histogram with degree values (0^0 to 180^0) on the 'X' axis, and their corresponding frequencies of angles on the 'Y' axis.



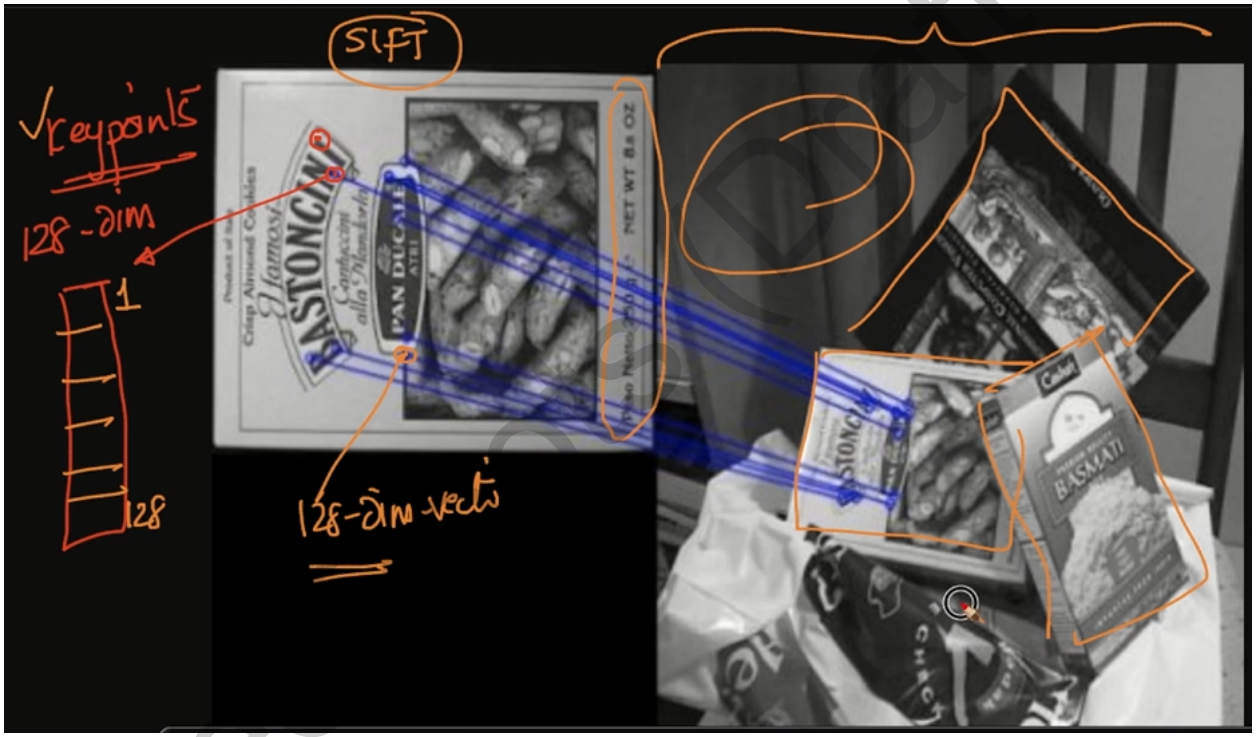
Note

While plotting the edge histograms, we have to also add the value '-1' on the 'X' axis, which indicates there is no edge present in the given cell. We then have to convert this histogram into a 182-dimensional vector(The values 0^0 to 180^0 and it also includes '-1')

45.6 Keypoints: SIFT

Scale Invariant Feature Transform (SIFT) is a very popular technique for detecting the objects in an image. It can also be used in featurizing the images.

The SIFT algorithm mostly detects the key points (which are mostly the corners) in a given image. For every key point, it creates a 128-dimensional vector for each of the key points. The collection of images that we use to compare the features for a new query point/image, is called **Database Image**. SIFT is extensively used in Image Search.



The images could be identified easily, even if the size of the image changes. This property is called **Scale Invariance**. Similarly, the images could be identified easily, if they are rotated. This property is called **Rotational Invariance**. SIFT technique possesses these two properties. If we need to compute the features using SIFT, we have to go with the **OpenCV** library, which has a Python interface.

Note

Key points in the SIFT technique are those pixels in which we have the objects. For example, if we have an image of a white wall with a door/window, then those pixels that cover the door/window are called **key points**, whereas the other pixels aren't taken into consideration.

Most image processing applications tend to use grayscale images over the color images, as the grayscale images tend to have much less information to process, and also it is simpler to handle the grayscale images when compared to the color images.

As we know there are 3 channels in an RGB image, and only one channel in a grayscale image, we end up having fewer parameters if we use grayscale images, when compared to the RGB images.

Q) Why do we get only a 128-dimensional vector as an output in SIFT when compared to any other number of dimensions?

Ans) SIFT is specifically a 128-dimensional vector that summarizes (or) describes a 16X16 window patch. The SIFT is obtained by dividing the 16X16 window into 4X4 bins. (Here each bin has the dimensions 4X4. When we divide, we get 16 bins)

Each bin has 8 orientation bins (or) channels. So the dimensionality becomes 4X4X8 = 128.

45.7 Deep Learning Features: CNN

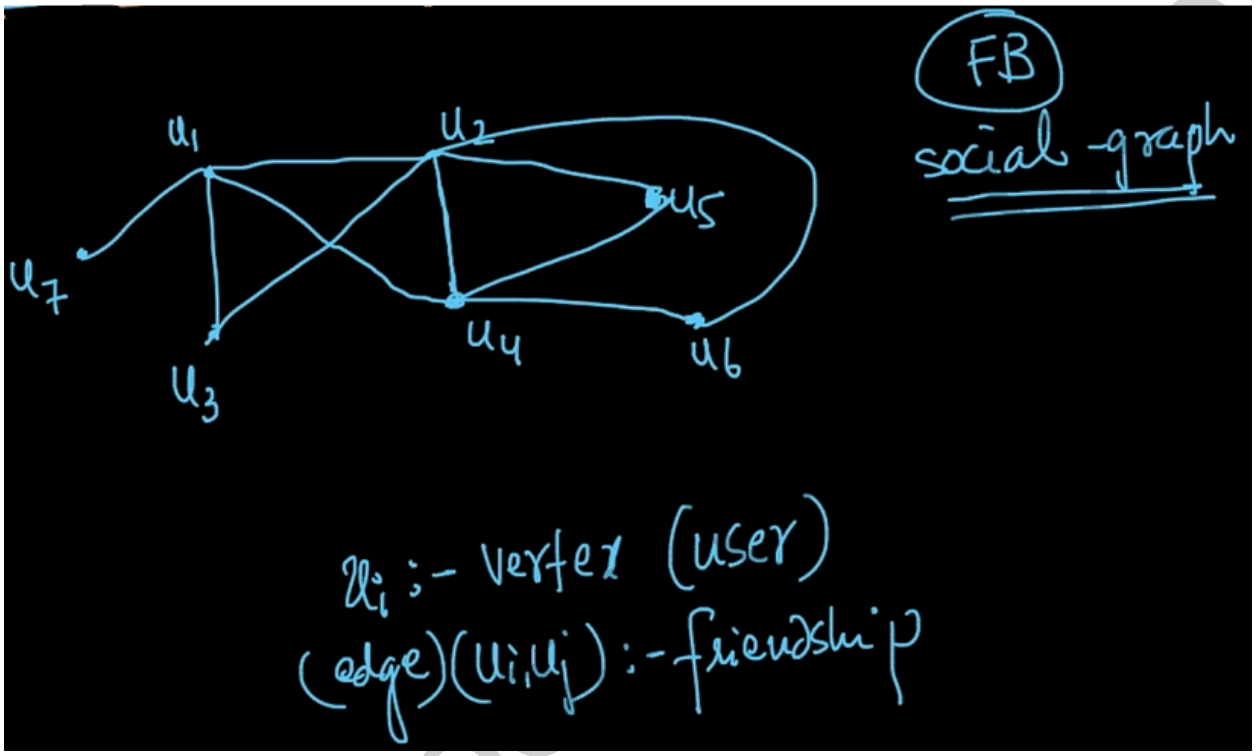
For featurizing the time series data, we have the LSTM technique. For featurizing the image data, we have the CNN technique. If we give lots of data to the CNN model, it will automatically detect the best featurization in the data. To solve a problem, the default technique is CNN (or) a variation of it in most of the image-based applications

45.8 Relational Data

If the data is stored in the form of tables, then we call the data as relational data. If the entire data is stored in multiple tables, then we have to pick the features that make some sense in the context of the problem we are solving, combine them and then perform data analysis. We also can do the same not only in SQL but also in Python (using Pandas library). The features that are picked in this way are called **Relational Features**, and these features will change when our problem changes.

45.9 Graph Data

Let us consider the problem of recommending friends for a user 'u_i'. If two users 'u_i' and 'u_j' are already friends, then we have to predict the label as '1', otherwise, we have to predict the label as '0'.



u_1, u_2, u_3, ….., u_7 are the **vertices** and the connections between these vertices are called the **edges**.

Let us assume we now have to recommend friends for 'u_4'. As 'u_4' is connected to almost all of them, we are left only with 'u_3' and 'u_7'.

Pair **Mutual Friends**

$(u_3, u_4) \rightarrow u_1$

$(u_4, u_7) \rightarrow u_1, u_2$

For example, we can design features like

$f_1 \rightarrow$ Number of Mutual Friends

$f_2 \rightarrow$ Number of paths distance (ie., the number of ways to reach from one vertex to another)

We can design many more features like this, and these features are called **Graph-Theoretic** (or) **Graph-Based** Features.

These kinds of graphs which have the users as the vertices and the friendships/connections as the edges are called **Social Graphs**.

For all the users 'u_i' and 'u_j', if there exists friendship/connection between 'u_i' and 'u_j', then $\mathbf{(u_i, u_j) = 1}$, otherwise $\mathbf{(u_i, u_j) = 0}$.

If the number of mutual friends (or) number of paths distance between 'u_i' and 'u_j' are more, then there is a high probability for these two users to become friends.

45.10 Indicator Variables

Converting the features indicator variables is one of the techniques of feature engineering. Deciding whether to apply this technique is problem-specific.

Example 1

If we have 'height' as a feature, and it is a real-valued feature, then

a) We can keep it as it is and go ahead with data modeling (or)
b) We can convert it into an indicator variable as
   if height > 150cm → class 1
   height <=150cm → class 0
   Here 'height' is a binary indicator variable. Mostly the indicator variables are binary.

Example 2

If a given feature is categorical, let's say the feature 'country' is a categorical feature, then it can be converted into an indicator variable as

```
if country == 'India' OR country == 'USA':
    return 1
else:
    return 0
```

Note

The usage of appropriate feature engineering techniques is problem-specific. Featurizing the data appropriately is a creative aspect of ML.

45.11 Feature Binning

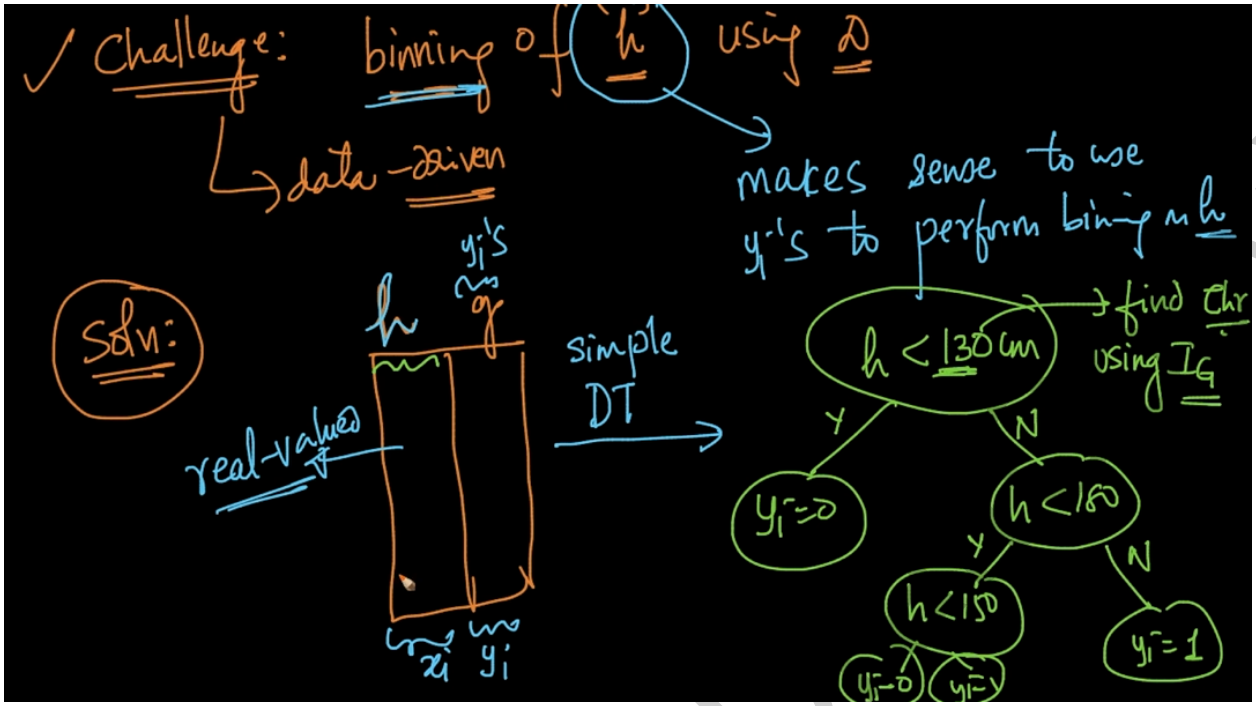
Feature Binning is a logical extension of the indicator variable technique.

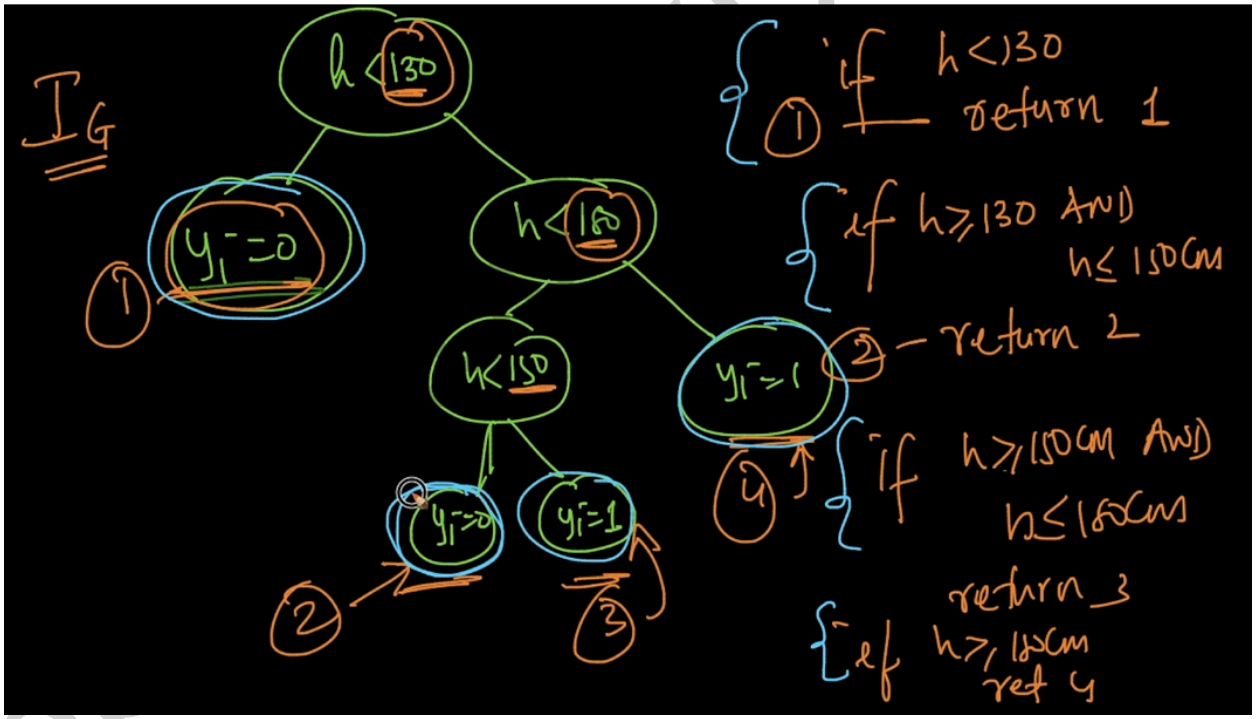
Example

Let us take the same height example that was discussed in the previous section, but the conditions get slightly changed when we apply Feature Binning.

```
if height < 120cm:
    return 1
if height < 150cm AND height >= 120cm:
    return 2
if height < 180cm AND height >= 150cm:
    return 3
if height >= 180cm
    return 4
```

Feature Binning is also called **Feature Bucketing**. It divides the available data values into buckets. In the above example, the values 120cm, 150cm, and 180cm are called the **Thresholds**. Finding out the right threshold is very problem-specific.


Challenge: binning of h using 2
data-driven
makes sense to use yi's to perform binning on h
Soln:
h
yi's
real-valued
xi
yi
simple DT
h < 130 cm
find Chr using IG
Y
N
yi=0
h<160
Y
N
h<150
yi=1
yi=0
yi=1


IG
h<130
h<160
yi=0
1
h<150
yi=1
2
yi=0
yi=1
2
3
4
{ 1 if h<130 return 1
if h≥130 AND h≤150cm
2 - return 2
if h≥150cm AND h≤160cms
return 3
if h≥160cm ret 4

45.12 Interaction Variables

Creating interaction variables is also a form of feature engineering.

Example

Let us consider the example of predicting gender on the basis of height, weight, hair length, and eye color.

If height<150cm AND weight<60kg, this is called the 2-way interaction feature.

The other examples of interaction are

a) height*weight
b) sqrt(height)*weight
c) (height<150cm) AND (weight<60kg) AND (hair length<75cm) → 3-way logical interaction feature.

Q) Given a task, how do we find good interaction features?

Ans) The first option we have is the Decision Trees. Let us assume we have 4 features: height, weight, hair length, eye color, and the output variable is gender.

Here we are coming up with the decision trees to find the new features, and these features will be useful to predict 'y'. Decision Trees in this context are used to find the interaction between the variables and figuring out the best set of features.

Here we are using a Decision Tree as a method to perform Feature Engineering.

Goal of Feature Binning/ Indicator Variables/ Interaction Variables

The goal is to engineer new features using the existing ones to better predict the class label. We typically add the new interaction feature to the original set of features and build a model. We typically never throw out the features and the data, unless we are really sure they are absolutely useless.

45.13 Mathematical Transforms

If ‘x’ is a single numerical feature, there is a type of feature engineering where we could apply mathematical operations on ‘x’, and this type of feature engineering technique is called **Mathematical Transform**.

Examples of Mathematical Transforms

a) $\log(x)$, e^x
b) $x^{1/2}$, $x^{1/3}$
c) x^2, x^3, x^4, …. (polynomial)
d) sin(x), cos(x), tan(x),....

The best feature transform for any problem is problem-specific and domain-specific.

Note

If ‘x’ follows a power-law distribution, then applying logarithm on ‘x’ makes sense. For tree-based algorithms like RF, DT, and GBDT, we need not perform the box-cox/log feature transform as these algorithms simply use thresholds to split the datasets.

In practice, RF and GBDT are very popular as they do not need simple feature transforms to work well.

45.14 Model Specific Featurizations

Example 1

Let us assume we have a feature 'f_1' which follows a power-law distribution, and we want to apply logistic regression (works on the basis of Gaussian Naive Bayes which assumes every feature in the given dataset follows gaussian distribution), in such a case, as Logistic Regression assumes the features to follow the Gaussian distribution, and the feature 'f_1' follows a power-law distribution, in order to make the logistic regression model work better, we have to transform 'f_1' from power-law distribution form to Gaussian distributed form, by applying logarithm on 'f_1'.

Example 2

Let us assume we have the features f_1, f_2, f_3, $y \in R$. From the domain knowledge, let's say 'y' is a linear combination of $\mathbf{f_i}$**'s**. Let it be in the form $\mathbf{y = f_1 - f_2 + 2f_3}$.

In this context, linear models like Logistic Regression would be appropriate. Models like DT, RF, etc couldn't work well on linear combinations.

Example 3

Let us assume from the domain knowledge, our class label 'y' is dependent on the interactions between the features 'f_1' and 'f_2'. For example, predicting the gender of a person depends on the interactions between height and weight. In this case, the behavior of these interactions among the men and the women is different and models like DT/RF/GBDT are the perfect models to be used to solve this problem.

In this case, linear models like Linear SVM, Logistic Regression would not work well and the interactions could be easily caught by DT/RF/GBDT.

45.15 Feature Orthogonality

The more different/orthogonal the features are, the better the models would be. Let us assume we have 3 features 'f_1', 'f_2' and 'f_3', and the response variable 'y'.

a) If each of the features 'f_1', 'f_2' and 'f_3' are highly correlated with 'y', and these features are highly correlated with each other, then combining these 3 features and using them, will have the least overall impact on the model we build to predict 'y'.
b) If each of the features 'f_1', 'f_2' and 'f_3' are highly correlated with 'y', and these features are not correlated with each other, then combining these 3 features and using them, will have a huge overall impact on the model we build to predict 'y'.

If we want to create a new feature 'f_4', then it must be highly correlated with the response variable 'y', but not correlated with the input features 'f_1', 'f_2', and 'f_3'. In such a case, adding 'f_4' to the model will have a huge impact.

Note

Two vectors are orthogonal to one another if their dot product is zero. A set of vectors is an orthogonal set if each distinct pair of vectors in the set have a dot product of zero. In two dimensions, it means the vectors are perpendicular to one another. Since the correlation of two random variables is zero, if the covariance is zero, according to this definition uncorrelatedness is the same as orthogonality. Independent variables are usually given as sequences of numbers, for which orthogonality is naturally defined by the dot product.

Note

We often use domain knowledge to design the features that are well correlated with the errors which often involves a lot of data analysis. Additionally, we can try and build feature interactions, mathematical transforms, etc which are well correlated with the errors to improve our model.

Q) How to design a feature 'f_4' that is highly correlated with 'y', and least/not correlated with 'f_1', 'f_2', and 'f_3'?

Ans) For all the points, we have to compute the error

$$e_i = y_i - y_i\hat{}$$

We have to design the feature 'f_4' such that it is highly correlated with the error (e_i). After designing 'f_4', we have to build a model with all the four features. This is similar to the concept of gradient boosting, where we compute a new feature (ie., error), add it to the model, and make predictions. We again compute the error, add it again to the model and make predictions. This way it continues.

But we should ensure that our model doesn't overfit, as the chances for overfitting are more in gradient boosting. In this scenario, we say 'f_4' is orthogonal to 'f_1', 'f_2', and 'f_3'.

45.16 Domain-Specific Featurizations

For example, if we want to predict a heart attack using the ECG data, it is always important to do some research and study the existing featurization techniques designed by the doctors/healthcare specialists.

For any domain, first of all, we have to do some research on the existing featurization techniques and try to build new ones on top of the existing ones instead of building everything again from scratch.

45.17 Feature Slicing

Let us consider the example of predicting the gender given the features are height, weight, hair length, eye color, country. Let's say the possible values for the 'country' column are 'India' and 'USA'.

If 80% of the data points have 'country' value as 'India', and the remaining 20% of the data points have 'country' value as 'USA', then if we build a model 'M' on it, then the model seems to perform better on data associated with 'India' in most of the cases and perform worse on data associated with 'USA' in most of the cases.

When we make predictions of the class labels on the test data, we do encounter a few errors in prediction. If the error rate in the case of points with minority 'country' value is very high when compared to the case with majority 'country' value, then the only strategy that could work is to split the train data 'D' into 'D_{Train}' and 'D_{USA}' and build 2 separate models.

Using this idea of slicing the data on the basis of the feature values, we need to build separate models for each feature value and compute the performance for each model.

In order to perform feature slicing, two conditions are to be satisfied.

a) Each category of a feature should have different behavior.
b) There should be a sufficient(decent) number of points in each slice.

Q) Is Feature Slicing applicable for the continuous numerical features also?

Ans) Yes, it can be applied to continuous numerical features. But feature slicing is proved to be more useful when applied to the categorical features.

Q) Do we have to apply feature slicing on every categorical feature in the dataset?

Ans) We do not separate the data for every categorical feature and create separate models. We'll check the error distribution for every value of the categorical feature. If the error distributions for each value are different, then we will go for the splitting of data, based on the categorical column values and we train multiple models.