07/04/2024 Chapter 3

Define the goal of steps in an ML project has the **most impact on being successful.**
Build the model is one step that spent the most time.

RSS/SSE error:

$$\text{Sum of residual squared } \sum_{i=1}^{n}(y_i - \hat{y_i})^2$$

SSR:

$$\text{Sum of regression squared } \sum_{i=1}^{n}(\hat{y_i} - \bar{y})^2$$

SST:

$$\text{Sum of variability squared } \sum_{i=1}^{n}(y_i - \bar{y})^2$$

Hence, $R^2 = 1 - \frac{RSS}{SST} = \frac{SSR}{SST}$, the bigger R^2 is, means that the model could explain more variability.

SAE error:

$$\text{Sum of absolute values } \sum_{i=1}^{n}|y_i - \hat{y_i}|$$

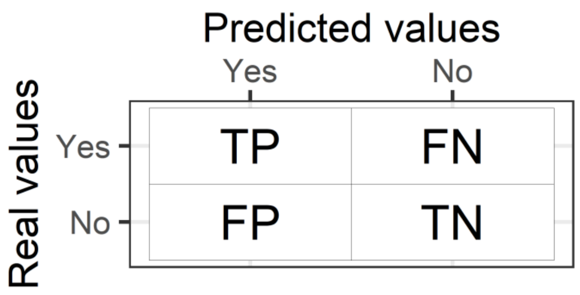
MSE error (L2 norm, outliers have **more** influence**):** 1/n * RSS
MAE (L1 norm, outliers have **less** influence**):** 1/n * SAE

Confusion matrix

*Where accuracy is not a good performance indicator, we evaluate **recall/sensitivity** ($\frac{TP}{TP+FN}$) & **precision** ($\frac{TP}{TP+FP}$) & **specificity** ($\frac{TN}{TN+FP}$), the overall indicator we called **F1 Score***

F1 score: $2 * \frac{Precision * Recall}{Precision + Recall}$

When prediction probability > threshold, model return **1**, means **positive prediction**, in the meanwhile, 1 means **true sample**, 0 means **false sample**.



Multi-class evaluation
The strategies we called **one-vs-rest** and **one-vs-one**

07/04/2024 Chapter 4

Issue when split train&test data set, **random shuffle** might introduce **bias**
Hence, train_test_split function in python/R is applied(stratify could be specify).
Another strategy to split dataset called **Stratified sampling**.

Try to split some numerical variables into **bins,** notice categorical variable is **ordinal** or **nominal**.
For unique variables of each row like matriculation number of students, removig that unless we can get some infors, like year they entered.

Scaling has **normalisation** and **standardisation** two ways.
Normalisation will rescale variables to 0-1 but it is **poor for outliers**.
Standardisation will ensure variables have mean equal to 0 and fixed variance but **bad for long tail variable** and it don't give a **specific bound** for variables that cause problems for some model.
PS: response variable should NOT be scaled.

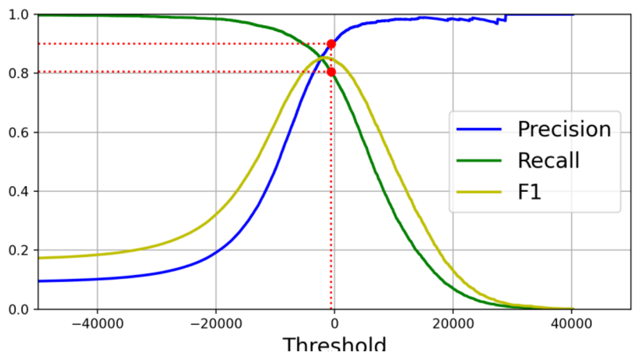
Using fit_transform() to train dataSet after we prepare the pipeline, and using transform() to test and validation dataSet.
If we use standardisation to scale train dataSet transform() to test dataSet, the mean won't be zero.

The good way to evaluate models's performance is using **K-fold cross-validation**(split training dataSet into K sets. using k-1 sets for train and left one for validation and do K runs). This strategy address the problems of **underfitting**(give few data for training, we got k-1 sets here) and **overfitting**(give few data to validation, we repeat K times here ensure a validation on the whole train dataSet). But K-fold cross validation increased computational cost.

Using grid research to fine tuning the final model(best Cross-validation/CV score).

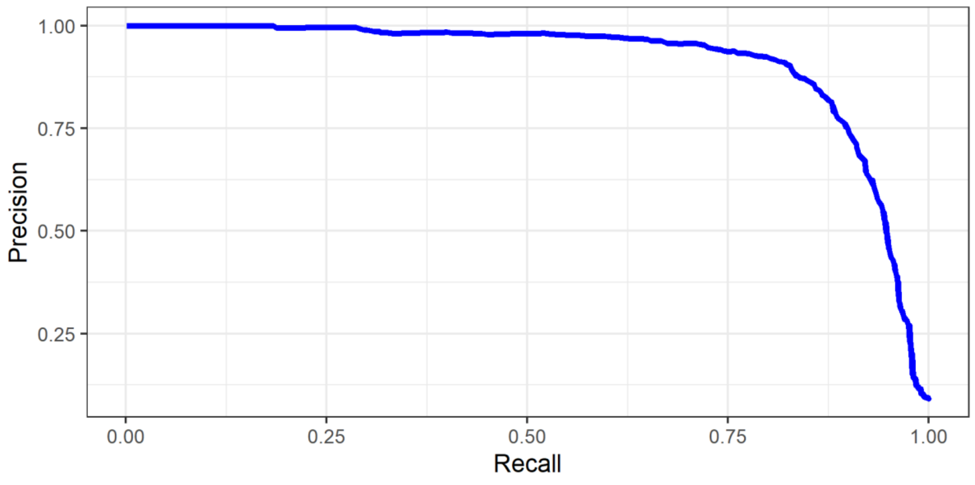
07/04/2024 Chapter 5

We could call **precision_recall_curve()** to return each instance's recall, precision and its threshold to determine its F/P, hence, we can plot Precision&Recall&F1 against threshold plot.



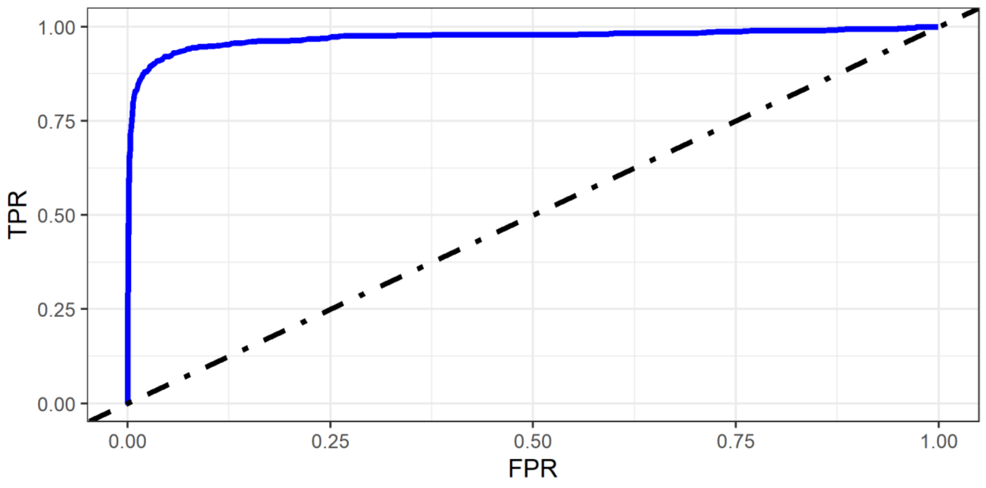
The **best threshold** is the interaction point between Recall and Precision, where the **maximum F1 score** shows.

We could plot Recall against Precision plot to choose the best classifiers.



Receiver Operating Characteristic curve(ROC curve)

It's similar to precision-recall curve, it's ***(1-Specificity)/FPR*** against ***Recall/TPR***, the area under ROC curve called **AUCOC**, ROC curve could be used to compare the classifiers too.



AUC also means: the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance.

Positive likelihood ratio: $LR_{+} = \frac{TPR}{FPR}$

Negative likelihood ratio: $LR_{-} = \frac{1-TPR}{1-FPR}$

Infors about EXAM:

In the exam you might be asked to

1. explain the types, theory and concepts of classification
2. produce and report on confusion matrices
3. analyse given confusion matrices
4. compare model outputs

5. build ROC curves from model data
6. analyse precision/recall or ROC plots

10/04/2024 Chapter 6

For estimate β_i in linear formula

$$y = \boldsymbol{X}\beta + e, \qquad \hat{y} = \boldsymbol{X}(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X})^{-\mathbf{1}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$$

Hence, $\beta = (\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X})^{-\mathbf{1}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$, for $y = X_0\beta_0 + X_1\beta_1 + \ldots + e$, we could estimate β by distribute a constant value to X_0,then solve input $(fixed\_X_0, X_1)$ and respone y, the key is left one variable X and response y while giving fixed values to remaining X

$$\beta_{0,1} = \text{solve } ((\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X})^{-\mathbf{1}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y})$$

Gradient for **MSE**

$$\nabla_\beta MSE(\beta) = \frac{1}{n} * (\mathbf{X}\beta - \mathbf{y}) * (\mathbf{X}\beta - \mathbf{y})^T = \frac{2}{n}\mathbf{X}^T(\mathbf{X}\beta - \mathbf{y})$$

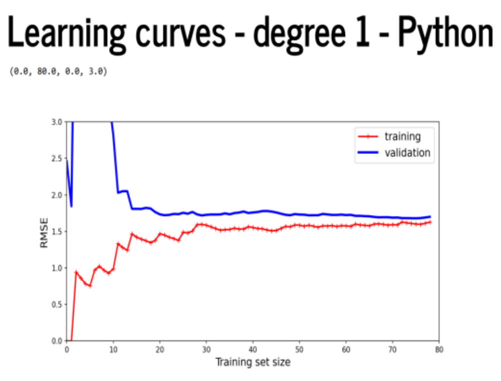
$$\beta = \beta - \text{learning rate} * \nabla_\beta MSE(\beta)$$

There are three ways for optimisation, **batch gradient descent** & **stochastic gradient descent** & **mini-batch gradient descent** the former need sum all instance's error(MSE) in one batch then calculate gradient, its cosume many time and might stuck in partial minimum, while later only randomly chose one instance in each batch(help **jump out** partial minimum).

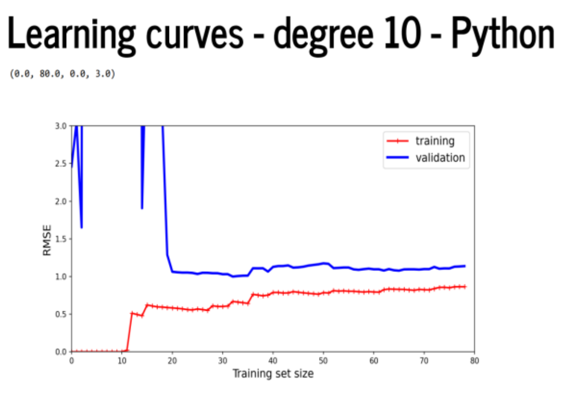
1. Batch converges straightforwardly
   but might be too expensive per iteration
2. Stochastic takes many iterations
   but each iteration uses very little memory
3. Mini-batch is a compromise

good performance on training data but bad cross validation error indicates **overfit**
poor performance on training data and cross validation indicates **underfit**

Evaluate Overfitting & Underfitting



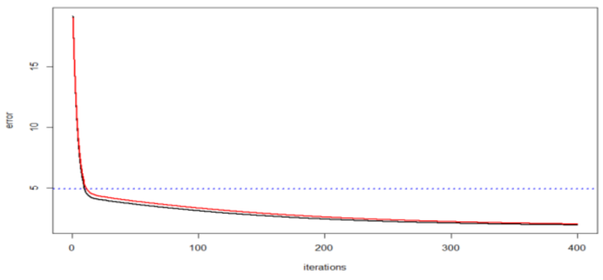
```
(0.0, 80.0, 0.0, 3.0)
```



```
(0.0, 80.0, 0.0, 3.0)
```

Left plot, add data wont decrease error, this plot shows the model is underfitting.
Right plot, the error on train set become small and the difference between train and validate set is bigger, which means that the model is overfitting.

Learning curves



Plot iteration against MSE error both on training and validation set will give a comprehensive indication that whether the model is underfitting or overfitting. The blue line is the MSE error comes from the simplest model, the current error is smaller than that means the model is not underfitting, the red line means the validation error, which slightly bigger than train set means that the model is not overfitting.

PS regularisation could be used for reducing overfitting.

Possible exam topics

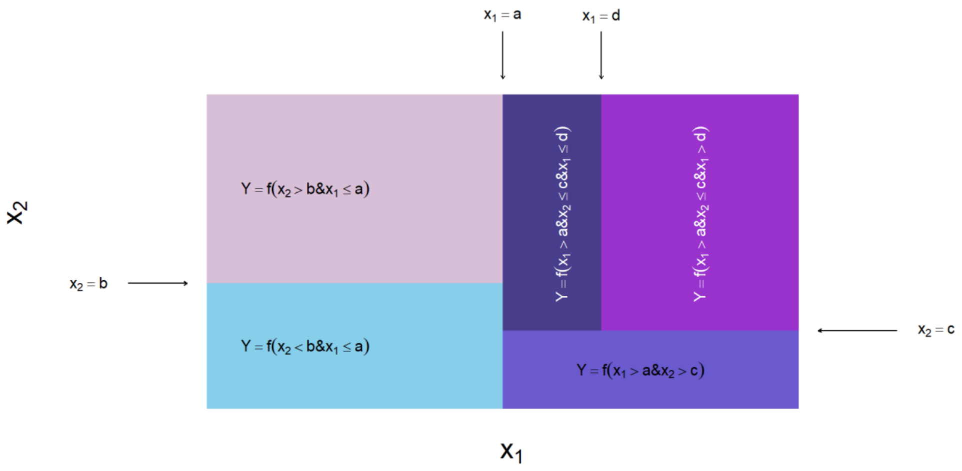
1. explaining and comparing concepts
2. interpreting charts of the types seen here

Exam won't include:

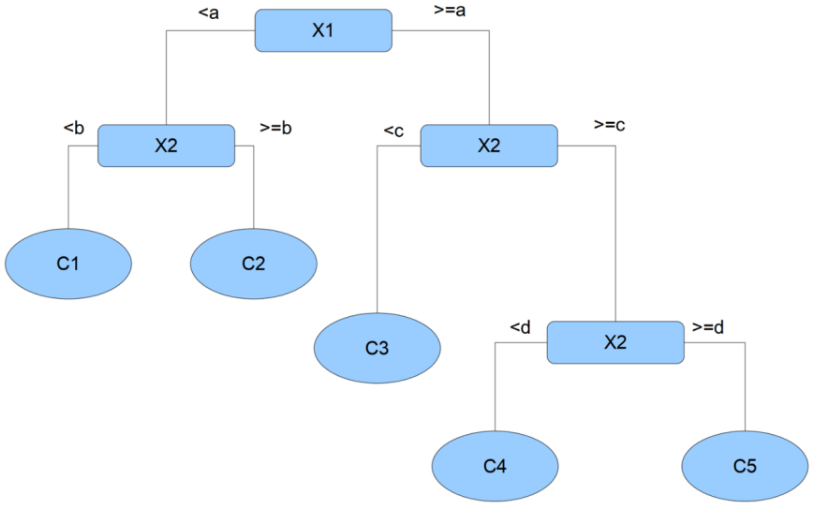
3. inverting a matrix
4. performing gradient descent

10/04/2024 Chatpter 7

How to split dataSet when get many dimensions(variables) for tree model.


$x_1 = a$
$x_1 = d$
X_2
$x_2 = b$
$Y = f(x_2 > b \& x_1 \le a)$
$Y = f(x_2 < b \& x_1 \le a)$
$Y = f(x_1 > a \& x_2 \le c \& x_1 \le d)$
$Y = f(x_1 > a \& x_2 \le c \& x_1 > d)$
$Y = f(x_1 > a \& x_2 > c)$
$x_2 = c$
X_1

These split points called nodes.



If tree model used for solve regression problem, the optimal constant for a leaf is the average of all ouputs comes from this leaf, if tree model used for solve classification problem, the optimal constant of a leaf is the most frequent category comes from that leaf. Decision tree belongs to supervised method.

Merit of tree model:

1. Accept both category of numerical data and don't centre or normalise them(not sensitive to outliers)
2. Nonparametric.
3. Automatically choose the variables and discover interactions among variables.

Purity: is the indicator used to evaluate the proportion of samples in each node.
Gini index(*G*): used for measure node impurity of classification problem, p_i means the proportion of category i in this node, k is the total number of categories. A node's gini generally lower but not always compare with its parent's node.

$$G = 1 - \sum_{i=1}^{k} p_i^2 \text{ (for one subcategory in attribute)}$$

If we want to calculate ***G/Gini gain*** for subcategory, we should always calculate the **base Gini(response)**, then using ***Base Gini****-1/P_subcategory1 *(G in subcategory: 1- …)-1/P_subcategory2*(G in subcategory:1-…) - …*

PS: Gini index for specific attributes is $\boldsymbol{sum\, proportion\, of\, each}\, [(1 - \sum_{i=1}^{k} p_i^2)]$, for example one attributes have three subcategories, and its binary classification, they have 5, 3, 6 samples respectively, there should calculate:

$$\frac{5}{14}*(1- \sum_{i=1}^{2} (\frac{a}{5})^2 + (\frac{5-a}{5})^2) + \frac{3}{14} *(1-\sum_{i=1}^{2} (\frac{b}{3})^2 + (\frac{3-b}{3})^2) + \frac{6}{14}*(1-\sum_{i=1}^{2} (\frac{c}{6})^2 + (\frac{6-c}{6})^2)$$

Entropy:

$$H(p_1, \dots, p_J) = \sum_{j=1}^{J} p_j \log_2 \frac{1}{p_j} = -\sum_{j=1}^{J} p_j \log_2\, p_j$$

Different tree have different ways to estimate **Mixed-up**:
In Regression tree,. Mixed-up is measured by **standard deviation/MSE**, and predict with the **mean**. In Classification tree, Mixed-up is measured by **amount of purity**, and predict with the **frequent.**

Standard deviation: $\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \overline{y})^2}$

MSE: $\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$

Infors about exam
How to calculate gini/entropy for given terms in tree model?
Can you draw a table to show that process?

10/04/2024 Chapter 8

Nonparametric bootstrapping: the bootstrapping is the resampling method to calculate a statistic(it's how we get confidence interval for ROC curve).

For ensemble methods, there are two families:
Average method(**indepentdent estimators**): forest of randomized trees, bagging methods. Boosting method(**sequential estimators**): Adaboost, gradient tree boosting, Xgboost, Catboost.

Bagging method: instead of training on different method, it trained on different subsets of trainSet(out of bag/OOB sampling, random choose subset **with replacement**, in this process, around 1/3 samples won't be selected, these samples called **out-of-bag samples**). For each subset grow a tree until overfitting, and prediction**(low variability**(predictor are less correlated due to use different training data) **but higher bias**(bootstrapping introdusce diversity in the subsets) than single tree model**)** is the most majority/average vote of trees. Repeat this process(grow tree and prediction) many times(bootstrapping).

Pseudo code(Bagging)

```
Use out of bag(OOB) sampling of dataset
Repeat the following process a few hundreds of times
Take samples with replacement
Grow a tree until expecting overfitting
Don't bother with pruning
Each tree give a classification(or value), the final prediction is the majority votes(average) from all trees.
```

Random forest is an ensemble of decision trees. But at each node of each tree, a sample of covariate is choosen **without replacement**. Random forest' error depend on two things:

correlation between any two trees in the forest; error rate of individual tree in the forest.
PS: higher P_RF increase both correlation and error rate, lower P_RF decrease both. **OOB error rate(average all trees' error rate)** used to select optimal P_RF.

Soft voting: sum probability of classifiers for each category and chose the maximum probability(might bigger than 1).
Hard voting: chose the majority vote as prediction.
Pseudo code

```
Choose $P_{RF}$(round to $\sqrt{P}$) features from total features $P$
Repeat the following process for a few thousand times
    1. Sample n rows from train set with replacement to get a subset with row n.
    2. Grow a tree using recursive partitioning
    3. At each node, sample $P_{RF}$ attributes without replacement and only test this for optimal splits(calculate gini impurity for current tree thereby ensure the attribute and best split, each attribute in $P_{RF}$ are calculated repectively).
    4. Keep growing tree until expecting overfitting
    5. Don't bother with pruning
Each tree give a classification or prediction, finally prediction is the majority vote(or average) from all trees.
```

Weak classifier: a classifier that perform only slightly better than chance.
Strong classifier: a learned classifier that correlates well with the true classifier.
Stump: a two-terminal-node tree; only two brunches, two leaves, it's a classic example of a weak classifier.

Adaboost
Pseudo code

```
1. Learn a basic classifier
2. Use this to increase the relative weight of incorrect predictions
    a) Focus on where the model perform poorly
3. Learn another classifier using these weights
    a) Update the weights
    b) Repeat until performance has improved
4. Predict class is the one with the majority of weighted votes from the ensemble.
```

PS: e_m means **measure error**, c_m means **classifier error**, stump or four/eight terminal classifier is most common used for Adaboost.
Rule of thumbs tell us:
Adaboost is difficult to overfit, stumps(weak classifier) works well, can try several and compare performance, each algorithm have many parameters that can be tuned and they have large effects.

Gradient boosting
Difference is look at residuals of the sequence of models
Fit a new model to residuals then combine with the original

The rules of thumb tell us:
Gradient boosting is easy to underfit and overfit casue too many or not enough predictors in the ensemble. And it have a poor choice of learning rate. Furthermore, it cannot handle missing value and unbalanced data.
PS (while bagging method in RF could maximise reduce the variance and overfitting, GBDT/gradient boosting maximise reduce the bias and underfitting.)

Xgboost(eXtreme Gradient Boosting)
Xgboost is basically build on (weak classifier)decision tree, it have several desirable features.

- clever penalization of trees.
- a proportional shrinking of leaf nodes.
- Newton Boosting(gradient and hessian matrix, while normal gradient boosting use gradient descent).
- automatic feature selection.
- breakthrought the limit of GBDT that build tree sequentially, it allows to build trees parallerly.

Formula of Xgboost's object:

$$Obj^{(t)} = \sum_{i=1}^{n} Loss\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + L_{1\ or\ 2}(f_t) + constant$$

Where $f_t(x_i)$ is the new tree generated for fitting the residual of last tree, the way of generating tree could be greedy algorithm, random chose a feature and list all structures of trees and find the minimum loss function value, and chose another feature repeat above process.

Key parameters that Xgboost used

- Booster: options(gbtree or gblinear), related to Loss function in formula.
- n_estimator: limit the maximum number of trees and iterations.
- learning_rate: normally set to 0.1.
- max_depth: maximum depth of trees.
- objective: options(reg:linear, reg:logistic, binary:logistic, multi:softmax, multi:softprob)
- eval_metric: options(RMSE, MAE, error(binary), merror(multi-class), AUC, mAP)

Catboost
Catboost is particularly designed for categorised features, its power origin from three technologies.

1. OTS (Ordered target statistics): distribute the unique order of categoriesd features, and sort the order by the target value, hence both the difference across class and intra class are considered, the overfitting is suppressed.
2. TAE (target-aware encoding): according to target encode categoried features dynamically, hence the relation between categories and target are more closed.
3. Handling missing value: regarding missing value as a new category, and calculate its information gain when build the tree, hence avoid the filling or delete operations.

Why and how boosting is Successful

1. It is model aggregation and reduces variance in the predictions.
    a) Hence useful for stabilising unstable classification method.
2. Weak learners typically biased
    a) boosting reduces bias as well as variance.

However, the boosting still lack of interpretability, although it could be extend to multiclass, the computational effort will increase sharply.
PS: boosting usually don't use bootstrapping sampling.

Infors

1. How boosting, bagging, random forests work - produce a schematic or pseudo-code.
2. Details of different variants (e.g. Adaboost, gradient boost)
3. What is meant by:
   a weak-learner,
   a stump,
   an Out-Of-Bag (OOB) estimate
4. Rules of thumb: how many iterations? what complexity of base learner should be used?
5. Where ensembles should work, where it probably won't
6. General pros/cons of the methods
7. Key hyperparameters for models

13/04/2024 Chapter 9

Too many dimension will casue hard to compute any solution and hard to find a good solution hence is called **curse of dimensionality**.
In high dimension, the distance become much far away between two nearby points, hence the prediction are less reliable.

We could find many points lie close at the lower level dimension, thereby use **PCA** to reduce dimension. The idea is find the axis that perserve the most variance in the data, then find the second highest axis…
Using **singular value decomposition(SVD)** find the principal components.

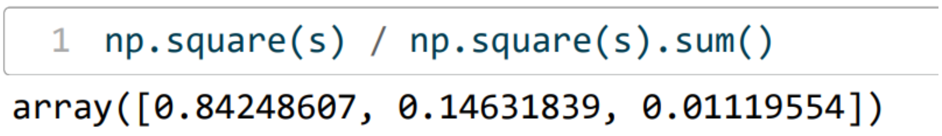
PCA only works when the data had already been centered(***X*_*centered*** = ***X*** – *mean(**X**)*).

Using np.linalg.svd(***X*_*centered***) to get ***U****(n*n)*, ***s****(n*p)*, ***Vt****(p*p).*

Vt *is consist of* $\{W_1, W_2, W_3, \ldots, W_p\}$

For column(dimension) d in ***X***, $\boldsymbol{X}_{d-projection} = \boldsymbol{X}W_d$

If we want to check the most important principal components, we could use $s^2/\sum s^2$



```
np.square(s) / np.square(s).sum()
```

```
array([0.84248607, 0.14631839, 0.01119554])
```

For three dimension dataset, we could see the first principal compoent could explain the most variance of the data. Ususally, we keep pc(principal compoents) those contain 95% variance of data totally.

This **PCA** process could easiy apply by using PCA package from sklearn.decomposition, pca.fit_transform(X) even conclude the centered step.

Print(pca.compoents_) could get ***Vt***.

Print(pca.explained_variance_ratio_) could get pc's variance proportion.

Furthermore, pca.inverse_transform() could reconstruct the X from X_svd.

13/04/2024 Chapter 10

Unsupervised learning

Clustering

1. Kmeans(hard-clustering, k-means++ still belong to hard clustering)
2. DBScan
3. Gaussian Mixtures(soft-clustering)
4. Agglomerative clustering

There are two ways to cluster: 1) hard clustering: assign each instance to the closest centre. 2) soft clustering: measure the distance of each instance to all centroids(according to each cluster's scale, distance, get weights(probability) of belonging to different cluster for each instance).

How to measure K-means: **inertia**(the sum of square distance between each instance and its closest centroid).

Silhouette score: a{mean(distance of this instance to others in same cluster)} – b{mean(distance of this instance to others in second nearest cluster)}/max(a, b). the range of silhouette score between -1 and 1. 1 means this instance is near at the center of current cluster, while 0 means near the boundary, -1 means this instance might was assigned to the wrong cluster.

When there are unbalanced instances and few labelled instance, using k-means as pre-process then do regression will help improve performance.

DBSCAN, compare with the hyperplane of k-means, DBSCAN could cope any shape while only need two parameters(eps and min_samples), besides that, it could identity and is robust to outliers. It work better when low density region separate the clusters.

Gaussian mixtures

Gaussian mixture could choose clusters's shape by parameter '**covariance_type**', and it could be used for anomaly detection(instance in low density regions can be consider as anomalies).

Model selection for clustering

We could not use inertia or silhouette score due to they both assume the clusters are spherical. Hence AIC or BIC are applied here.

$$BIC = \log(m)p - 2\log(\hat{L})$$

$$AIC = 2p - 2\log(\hat{L})$$

Where m is the number of instances, p is the number of parameters of the model. $\hat{L}$ is the maximised value of the likeihood function of the model.

15/04/2024 Chapter 11

SVM is about finding a line that is useful for predictions(the one furthest away from data). The point closest to the line are called support vectors. The essence is to use the Langrangian dual algorithm to solve the convex quadratic programming problem(linear, if non-linear use kernel function/mapping to higher hyperplane).

Our objective function is find the hyperplane where(we have many dimensions hence we have multi-hyperplane) have the maximum margin/distance between suppot vector and hyperplance.

$$max(\frac{1}{\| w \|}) \text{ , where } y(w^T x + b) - 1 \geq 0$$

Hence, it could be changed to covex quadratic programming problem:

$$min \frac{1}{2} \| w \|^2$$

PS (if data points seems couldn't split in current dimension, it always works when we mappint it to higher dimension, and some outlier data point will extremely influence splitting hence introduce a slack feature ξ to allow loose splitting.)

SVM Don't give probabilities only return predictions. Try extened way to help svm get probabilities is very slow and don't always match with predictions. Besides that, svm is default **binary.**

The closest distance from hyperplane to positive point is d_+, the closest distance from hyperplane to negative point is d_-.

This distances is called the **margin.**

Hard margin(linearly separate, all data is outside the margin)

- Svm are sensitive to the **scale of the features** and **outliers**. Hence centre and scale the data when using svms.

Soft margin(data could be within the margin and even in wrong side)

After change parameter C of regularization,

- Low C leads to wider margins, more violations, morelikely to underfit.
- High C leads to narrow margins, fewer violations, more likely to overfit.

Svm could also be used for regression. The aim change from fit the largest possible margin between classes to fit as many as instances within the margin.

Kernel technique could avoid calculate the intermediate features(Essentially used to find the inner product of two functions).

- If number of features and number of samples are nearly same, using Linear Regression or SVM with linear kernel.

- If number of features are few and number of samples are ok, using SVM with gaussian.
- If number of features are small and number of samples are big, adding some features manually and use LR or SVM with linear kernel.

Difference between LR and SVM:

- Both are classifier and could do binary classification.
- Both are supervised algorithm.
- Both are discriminative model don't care how the data generated and focus on difference among data.
- Both could handle non-linear question.
- LR belong to parameter model while SVM are non-parameter.
- Loss function are different: LR use logistic loss and SVM use hinge loss.
- LR is simple to calculate hence suitable for big dataset while SVM is complex to calculate hence suitable for small dataset.

Difference between LR and RF:
The fitting curve of RF is stepped, the LR is smooth curve.
RF have best classificate ability benefit of information gain(gini-purity)
LR suitable for sparse high dimension features hence binning would benefit LR while RF is suitable for dense high dimension features.

15/04/2024 Chapter 12

Artifical neural networks
Traditional NN have a pyramid architecture, it require numeric data, missing data, unscale would cause problem, hence cleaning and imputation, scaling data is important.
Data = batches * epochs
NN compoents: input layer, hidden layer, output layer, bias node.

The activate functions contains: ReLU(max(0, x)), sigmoid($\sigma(x) = \frac{1}{1+e^{-x}}$), tanh.

ReLU usually applied in hidden layers, every output frrom hidden layers should go through ReLU function then pass to outputs layers, sigmoid usually applied in outputs layers to get probabilities.
Back propagation obey the chain rule, which aim at

- Calculate how much each connection in the output layer contributes to the error.
- Calculate how muchb each connection in each previous layer contributes to the error.

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}, \ w_i = w_i - \eta \frac{\partial L}{\partial W_i}$$

Problems of neural networks: stuch at a local minimum, might converge very slow due to gradient already become very small, specific activate functions would like to affect performance.

Infors about exam

1. Equations for a single neuron: $y = f(\sum_{i=1}^{n} w_i x_i + b)$
2. How to draw a neural net

3. **The principles of back propagation**

   The back propagation contain four steps:

   1) Forward propagation, record results of each layers between inputs to outputs.
   2) Loss function: calculate the loss base on real value and outputs.
   3) Back propagation: according to loss generate gradient of each weight base on chains rules.
   4) Update the weighs base on step and learning rate and previous weights.

4. Be able to describe different techniques to deal with over fitting
   1) Regulization.
   2) Dropout
   3) Early stopping.
5. How to deal with vanishing / exploding gradients
   1) Using ReLU avoid gradient close to zero.
   2) Initialise the original weights using Glorot make sure they are in scale.
   3) Using normalization on each hidden layers.
   4) Gradient crop, avoiding exporation of gradients.
6. Describe different types of optimizers
   a) the general intuition or principles of how they work
   b) you do not need to know all the equations

10/05/2024 AB Test

Normal Steps of AB test

1. Ensure the controlled covariable (experimented against controlled, choose one each time)
2. Ensure the response variables (click rate or re-view rate)
3. Ensure the sample size and period of the experiment (we expect to see is the minimise dataSize)
4. Split the sample in order to keep the random and approximately normal distribution (avoid Simpson's Paradox-where the inappropriate group tends to lead completely conversed trend in group when compared with the whole data)
5. Pre-grey test by using small sample size (make sure choosed controlled covariables are reasonable)
6. Test statistic check(Student test statistic)

In step 2: the response variable(observed indicator) are split into two groups:

- Absolute indicators (could be calculate by singluar covariable such as DAU, using T test)
- Rate indicators (should be calculated through multiple covariables such as re-buy rate, using Z test)

PS (when compare multi-group's test, using Anova(variance analysis, based on F test))

When different response varible are choosed, the ways to calculate the minimise sample size in step 3 are different.

Calculate the sample size:

$$\mathrm{N} = \frac{(z_{1-\frac{\alpha}{2}} + Z_{1-\beta})^2 * \sigma^2}{\delta^2} \approx \frac{8 * \sigma^2}{\delta^2}$$

Where α(first error) and β(second error) usually set to 0.05 and 0.2 respectively, hence $z_{1-\frac{\alpha}{2}}$ = 1.96 and $Z_{1-\beta}$=0.84, whileδ is the difference between controlled response variable and experimented response variable.

- When the response variable belong to Absolute indicator,

$$\sigma^2 = \frac{2^* \sum_1^n (x_i - \overline{x})^2}{n-1} = 2 * sqrt(var)$$

$$where\ n\ is\ experiment\ sample\ size\ and\ x\ is\ the\ expected\ value\ of\ samples$$

- When the response variable belong to Rate indicators,

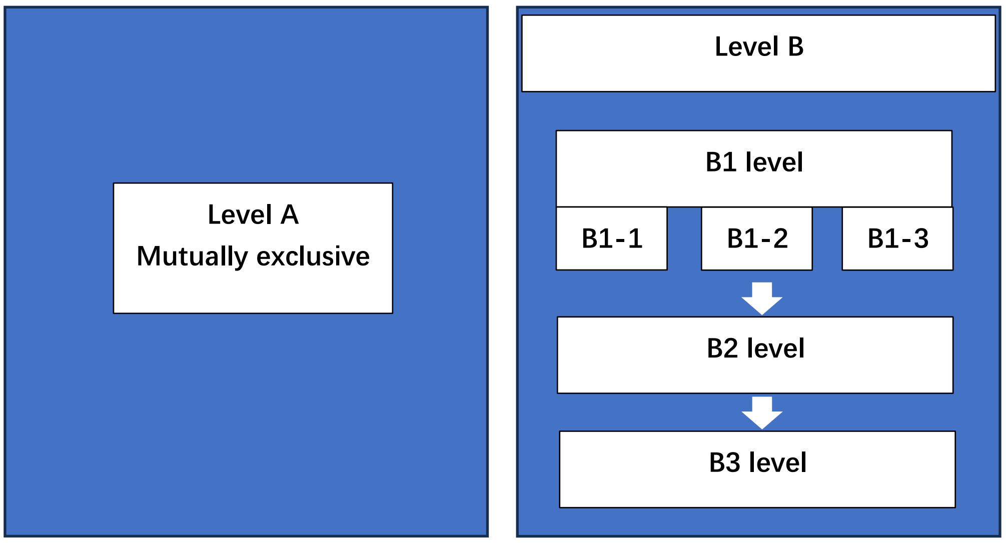
$$\sigma^2 = P_A(1-P_A) + P_B(1-P_B)$$

$$where\ P_A\ and\ P_B\ are\ the\ response\ variable\ of\ experimented\ and\ controlled\ group\ respectively$$

Simpson's paradox

Make sure all covariables are equally split expect for response variable, there are three ways to split the sample.

1. mutually exclusive: each sample only allow to stay in one group, hence the total number of groups are N/each group's size
2. Orthogonal: each sample allow to stay in one group in each level, while we must provide no relationship saw between with levels.
3. mutually exclusive & Orthogonal: if some levels are correlated and some levels are independent, we apply mutually exclusive to correlated levels while apply Orthogonal to independent levels.



Where level A and B are correlated and B_i in level B are independent, in each B_i level, the sample are used repeatly. The changes in each level B_i have equal influence to other levels in level B and Level A.

04/09/2024 method of binning of continuous variables

Merit of binning

1. When variables contain some outliers, binning would help variable be more robust.
2. Linear link model like *LR* would fit better due to binning bring non-linear informations.

Where should not use binning: Model uses weak classifier like *GBDT* would converge slowly due to binning mapping variables to sparse space and it tends to lead to overfitting. Hence if the model accepts continuous variables don't need to be binning.

Three ways of best binning

a) Optimal binning based on CART(classification and regression tree)

CART algorithm base on recursion of minimum gini-index chose the optimal splitting point to generate stumps. The process of CART binning are:

1. Sort the given variable V.
2. Calculate the median of adjacent elements from the first element to last element.
3. Chose the median that have greatest decrease of gini-indexs to generate current stump.
4. Repeat step 1, 2, 3 in two branchs of stump.

b) Optimal binning based on Chi-square distribution

The Chi-square value are calculated by below formula, where A indicates the actual frequency and E indicates expected frequency. Binomial form is given here too, m means the observed number of successes in N trial, p is the probability of success, $q = 1 - p$.

$$\chi_k^2 = \sum_{i=1}^{k} \frac{(A_i - E_i)^2}{E_i} = \frac{(m - Np)^2}{Npq}$$

The chi-square table is provided here. Df= (Nrow-1) * (Ncol-1). If p-value related to chi-square value is bigger than 0.05 or 0.1 we could say that two group of data have no difference further could be combined.

Degrees of freedom (df)	χ^2 **value**[20]										
1	0.004	0.02	0.06	0.15	0.46	1.07	1.64	2.71	3.84	6.63	10.83
2	0.10	0.21	0.45	0.71	1.39	2.41	3.22	4.61	5.99	9.21	13.82
3	0.35	0.58	1.01	1.42	2.37	3.66	4.64	6.25	7.81	11.34	16.27
4	0.71	1.06	1.65	2.20	3.36	4.88	5.99	7.78	9.49	13.28	18.47
5	1.14	1.61	2.34	3.00	4.35	6.06	7.29	9.24	11.07	15.09	20.52
6	1.63	2.20	3.07	3.83	5.35	7.23	8.56	10.64	12.59	16.81	22.46
7	2.17	2.83	3.82	4.67	6.35	8.38	9.80	12.02	14.07	18.48	24.32
8	2.73	3.49	4.59	5.53	7.34	9.52	11.03	13.36	15.51	20.09	26.12
9	3.32	4.17	5.38	6.39	8.34	10.66	12.24	14.68	16.92	21.67	27.88
10	3.94	4.87	6.18	7.27	9.34	11.78	13.44	15.99	18.31	23.21	29.59
***p*-value (probability)**	0.95	0.90	0.80	0.70	0.50	0.30	0.20	0.10	0.05	0.01	0.001

The process of chi-square binning are:

1. Sort the given variable V and regard every element as a group.
2. Calculate the chi-square value on adjacent group from the first group to last group.
3. Merge the two group that have lowest chi-square value and smaller than 0.1.

4. Repeat step 1, 2, 3 until we get enough groups.

c) Optimal binning based on KS

The process of optimal KS binning:

1. Sort the given variable V.
2. Calculate KS on every element and chose the element have biggest KS.
3. Repeat step 1, 2 until KS is smaller than a threshold. (KS will keep decrease due to binning)

How to evaluate the effect of binning?

By comparing the value of IV.