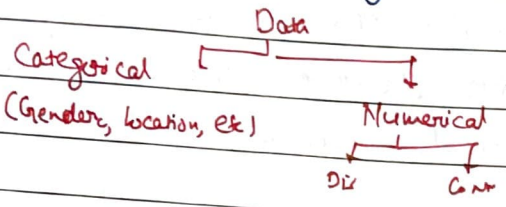
STATISTICAL METHODS FOR DECISION MAKING.

Descriptive: - Concerned with data summarization, Graphs/Charts and tables.

Inferential: — Talk about a population parameter from a sample, point estimation and hypothesis testing.

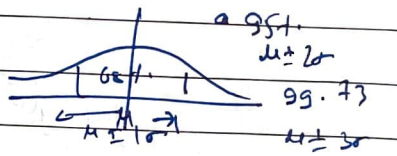


Measure of central tendency:

1. AM: $\bar{X} = \Sigma x/n$, affected by extreme values, outliers
2. Median: $(n+1)/2$, great resistance to outliers (Can combine medians or mode of several groups)
3. Mode: (Value which occurs most often).

Measure of dispersion: How large the spread is around central tendency.

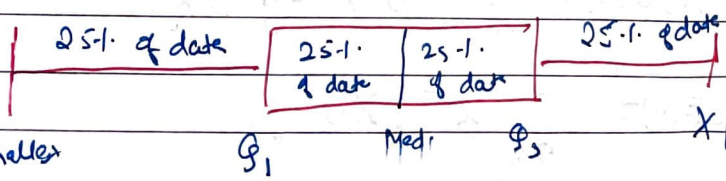
1. Range: - $X_{max} - X_{min}$
2. IQR: - (less effected by outliers): $Q_3 - Q_1$
3. SD: - $\sqrt{Var}$



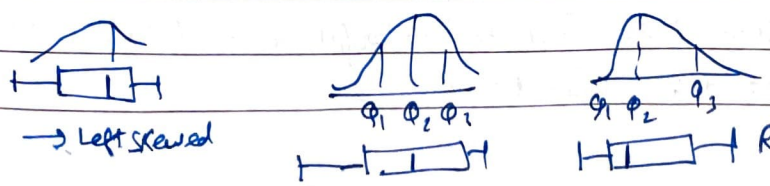
→ CV: (Coeff of variation) : CV2 S.D / μ

Five number summary: - $X_{smallest}$, First Quartile (Q_1), Median (Q_2), Q_3, $X_{Largest}$

Left skewed	Symmetric	Right skewed
Median − $X_{smallest}$	Median − $X_{smallest}$	Median − $X_{smallest}$
>	≃	<
$X_{largest}$ − Median	$X_{largest}$ − Median	$X_{larg.}$ − Median.
Q_1 − Med − Q_1	Median − Q_1	Median − Q_1
>	≃	<
Q_3 − Med	Q_3 − Med	Q_3 − Med.



Outlier if
1.5 times
IQR



→ Covariance: Measure the strength of linear relationship b/w two numerical Var (X and Y).

$$Cov(X,Y) = \sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y}) / n-1$$

(Doesn't measure the relative strength)

→ r = Coeff of correlation: $r = Cov(X,Y) / S_x S_y$ $\quad -1 \le r \le 1$

Gives relative strength of linear relationship between two numerical variables.

PYTHON NOTES

→ Linear Reg:- $\hat{y} = h_\theta(x) = \theta^T \cdot x$

θ Contains θ_0 = bias term and feat weights θ_1 to θ_n.

x = instance feat vector: x_0 to x_n where x_0 is equal to 1.

→ Cost function:- $MSE(\theta, h_\theta) = \frac{1}{m} \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)})^2$

↳ soln:- $\hat{\theta} = (X^T X)^{-1} X^T y$

$y \in (y^{(1)}$ to $y^{(n)})$, $\hat{\theta}$ = value of θ that min^n cost function.

→ Scikit has Linear Reg.

→ scipy.linalg.lstsq() [least sq).
↳ calculates $\hat{\theta} = X^+ y$, X^+ is pseudo inv. of X

→ (The Moore Penrose inv).

→ np.linalg.pinv(X_b).dot(y) [directly calculate pseudoinv)

pseudo inv is basically SVD calculation. $U \Sigma V^T$

$X^+ = V^T \Sigma^+ U^T$, Σ^+ = sets all other values to zero than threshold and non zero by their inv. and tr.

* $(X^T X)$ has an issue it may or may not work due to invertibility such that m<n but pseudoinv will always work.

SVD $O(n^2)$.

→ $\nabla MSE(\theta) = \frac{2}{m} \sum_{i=1}^{n} (\theta^T x^{(i)} - y^{(i)}) x_j^{(i)}$

$\nabla_\theta MSE(\theta) = \frac{2}{m} X^T (X\theta - y)$

→ Adding More feat to underfitting model will not help Use diff model.

In case of overfitting add more data

Model complexity ↑ Var ↑ overfitting
↓ Var ↑ underfitting

→ Ridge (L2) :- $\alpha \sum_{i=1}^{n} \theta_i^2$ = → add to cost function only.

d.s hyp:- , $\alpha = 0$, it's linear, $\alpha \simeq \alpha$, all are weights → 0. flat line going through mean.

$J(\theta) = MSE(\theta) + \frac{1}{2} \alpha \sum_{i=1}^{n} \theta_i^2$

→ Is the more the value:

$\|V\|_k = (|v_0|^k + |v_1|^k + |v_2|^k + \cdots + |v_n|^k)^{\frac{1}{k}}$

more k↑ neglects small value and put emphasis on outlier so RMSE is sensitive to MAE.

∴ Standard scaler should be used befor L_2 or Ridge Norm.

→ $\hat{\theta} = (X^T X + \alpha A)^{-1} X^T$

A is (n+1) x (n+1) identity Mat.

↳ Uses Andre Louis Cholesky method.

→ Lasso (L1) :- $J(\theta) = MSE + \alpha \sum_{i=1}^{n} |\theta_i|$

[It tends to eliminate weight of least imp feat make it 0. i.e all high degree poly are zero, — Automatically performs feature selection and output a sparse.

→ Elastic Net (Middle ground) :- r = mix ratio.

$J(\theta) = MSE + r\alpha \sum_{i=1}^{n} |\theta_i| + \frac{1-r}{2} \alpha \sum_{i=1}^{n} \theta_i^2$

↳ Preferred over Lasso.

→ Logistic or Logit Reg: Binary classifier. , $\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$

$\hat{p} = h_\theta(x) = \sigma(X^T \theta)$ sigmoid. $\frac{1}{1+e^{-t}}$ algn

$c(\theta) = \begin{cases} -\log(\hat{p}) & \text{if } y = 1 \\ -\log(1-\hat{p}) & y = 0 \end{cases}$

→ $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(\hat{p}^{(i)}) + (1-y^{(i)}) \log(1-\hat{p}^{(i)}) \right]$

↳ Convex function.

→ Softmax:- $S_k(x) = x^T \theta^{(k)}$

$$p_k = \sigma(s(x))_k = \frac{\exp(s_k(x))}{\sum_{j=1}^{k} \exp(s_j(x))} \qquad k = \text{no. of class}$$

$s(x)$ = Vector containing the scores.

$$\hat{y} = \arg\max_k \sigma(s(x))_k = \arg\max_k s_k(x) = \arg\max_k \left((\theta^{(k)})^T x\right)$$

→ Cross entropy: $-\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{k} y_k^{(i)} \log\left(\hat{p}_k^{(i)}\right)$

$y_k^{(i)}$ → target prob. i^{th} in k class.

$$v_\theta = \frac{1}{m} \sum_{i=1} \left(\hat{p}_k^{(i)} - y_k^{(i)}\right) x^{(i)}$$

* SVM:- Sensitive to feature scale.

→ Hard Margin:- Sensitive to outliers and works with only linearly sep. data. low value of $C \propto \frac{1}{\text{slack}}$ in scikit.

C is regularisation factor.

→ Also instead of using LinearSVC, SGDC can be used with loss = 'hinge', $\alpha = 1/(m*C)$ applies to Stochastic GD. to train SVM classifier. (used for large datasets)

Non-linear SVM: Gaussia: $e^{-\gamma \|x - l\|^2}$ (γ and C hyp)

Complexity: ϵ = tol in scikit- only SVC supports kernel trick.

		Scaling	Kernel
LinearSVC	$O(m \times n)$	✓	X
SGD Class.	$O(m \times n)$	✓	X
SVC	$O(m^2 \times n)$ to $O(m^3 \times n)$	✓	✓

Regression: ϵ is margin insensitive to data

→ $\hat{y} = \begin{cases} 0 & \text{if } w^T x + b < 0 \\ 1 & \text{if } w^T x + b \geq 0 \end{cases}$ $= w_1 x_1 + w_2 x_2 + \dots + w_n x_n$

$$\min_{w, b} \frac{1}{2} w^T w \simeq \frac{1}{2} \|w\|^2$$

Subject to: $t^{(i)} (w^T x^{(i)} + b) \geq 1$ for $i = 1, 2, \dots, n$

Softmargin:

$$\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi^{(i)}$$

$$\text{Subj to } t^{(i)} (w^T x + b) \geq 1 - \xi^{(i)}, \quad \xi^{(i)} \geq 0, \quad i = 1 \dots n$$

Both are convex quadratic optimization (Eg Richard Brown in Boyd)

→ Primal normal

dual for svc.

→ Kernel: — in ML means a function capable of computing the dot product $\phi(a)^T \phi(b)$ based only on original vector $\vec{a}$ and $\vec{b}$, without having to compute the transformation ϕ.

→ Linear: $K(a,b) = a^T b$

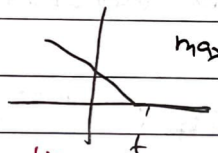
Poly: — : $K(a,b) = (\gamma a^T b + r)^d$

Gaussian RBF : $K(a,b) = \exp(-\gamma \|a-b\|^2)$

Sigmoid : $K(a,b) = \tanh(\gamma a^T b + r)$

→ online SVM: (online means learning incrementally, typically as new instances arrive.

$\max(0, 1-t)$ = hinge loss.



$$J(w,b) = \frac{1}{2} w^T w + C \sum_{i=1}^{m} \max(0, 1 - t^{(i)} (w^T x^{(i)} + b))$$

* Decision Trees: (both class. and reg. task)

Adv:- Less data prep, Almost no scaling or centring.

Gini calculates impurity, pure => gini = 0

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$$

→ CART is used by Scikit. they have always 2 child unlike older ones ID3 which have more than 2.

↳ Decision tree is called white box Model because it's easy to interpret unlike Neural and Random forest.

Depth = $\log_2(m)$

→ CART (Classification and Regression tree)

$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$

m = no of instances, G = Gini or Measure of impurity

→ Greedy algo. ~~(O(exp(m)) time~~.

→ $O(\log_2(m))$ nodes. , Also gini is faster to compute than Entropy.

→ Non-parametric model: [∵ no of parameters is not det. prior to training.

→ max_depth is hyp. and used as regularisation too.

→ Cart for Rg. $J(k, t_k) = \frac{m_{left}}{m} MSE_{left} + \frac{m_{right}}{m} MSE_{right}$

where $MSE = \sum_{i \in node} (\hat{y}_{node} - y^{(i)})^2$

$\hat{y}_{node} = \frac{1}{m_{node}} \sum_{i \in node} y^{(i)}$

Cons: → They love to be orthogonally decision boundary, makes sensitive to training set rotation.

→ Small variation will cause changes.

Overfitting ↓ max_depth, independent of Scaling.

* Ensemble Learning and Random Forests:—

→ Due to laws of large number, work best on methods that are independent of each other

→ Voting Classifier [%

Bootstrap agg.

Bagging:- Sample is performed with replacement, this is called bagging. and Pasting when sampling is performed without replacement. [Pg 193 diagram].

Allows training instances to be sampled several times for same predictor

→ Predictors can be trained in parallel, and prediction can be made in parallel.

Random Patches Random Subspa

pasting.

→ Bagging ends up slightly higher bias than pasting, but small var. & overall bagging results in better model.

However ultimately cross Val should be used to test if bagging or boosting is efficient.

→ Out of bag eval. Due to bootstrapping, with rep. 63.1% are sampled for each predictor ($1-\exp(-1) \approx 63.21\%$) Remaing 37% are not sampled called out-of bag (OOB) instances.

Random Patches Random Subspaces → max_feat, bootstrap_features are two hyperparameter. → Good for high-D ip. Sampling both training ins. and feat. is called Random patches.

Random forest: Ensemble of D.T. generally trained via bagging method. RandForest Clas – hyperparameter is DecisionTree Cla.

→ Instead of searching for the best feat. when splitting a node it searches best features among a random subset of feat. → Results in greater div → Greater bias → Low Var. → Overall better model.

→ Extra tree :- Feat & Feature Imp :- using feature imp. var.

→ R.F are very handy to get quick understanding of what feat. actually matter, good for feat. selection.

Boosting: Ensemble method that combine several weak learners into a strong learner. → Train predictor sequentially.

→ Ada Boost: [Pg 200]. Drawback:- Not be parallesid. Does not scale as well as Bagging or pasting.

$$r_j = \frac{\sum_{\substack{i=1 \\ \hat{y}_j^{(i)} \neq y^{(i)}}}^{m} w^{(i)}}{\sum_{i=1}^{m} w^{(i)}}$$

$$\hat{y}(x) = \underset{k}{\operatorname{argmax}} \sum_{\substack{j=1 \\ \hat{y}_j(x)=k}}^{N} \alpha_j$$

Final

Predictor Weight: $\alpha = \eta \log \frac{1-r_j}{r_j}$

Weight update

$$w^{(i)} \leftarrow \begin{cases} w^{(i)} & \text{if } \hat{y}_j^{(i)} = y^{(i)} \\ w^{(i)} \exp(\alpha_j) & \text{if } \hat{y}^{(i)} \neq y^{(i)} \end{cases}$$

Split has SAMME.

- overfits than Pg & learning rate

(iii) Gradient Boosting: Works by sequentially adding predictor to an ensemble, each one correcting it's pred.

Uses Residual error made by pre. unlike ada boost. tweaks instance weight of predecessors.

→ import XGboost. (optimised implementation of GB Gradient boosting)

Stacking: [Pg 206].

DIM. REDUCTION:- (i) Projection (ii) Manifold learning.

(iii) PCA. → AIM FINDING MAX^m VAR

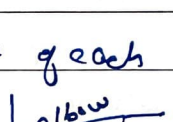
→ Works on SVD:- $U \Sigma V^T$ contains unit vect that defines all PC.

$$V = \begin{bmatrix} | & | & | \\ c_1 & c_2 & c_n \\ | & | & | \end{bmatrix}$$

(Data should be centered around origin)

→ $X_{d\,proj} = X W_d$. , to find value of each var ratio.

explained var ratio.



→ Use randomized PCA for faster one $O(m \times d^2) + O(d^3)$ or rather than $O(m \times n^2) + O(n^3)$

→ Incremental PCA to solve instead of whole dataset Split into mini batch and feed IPCA algo one by one - mini batch.

→ Kernel PCA:-

LLE (Locally linear embedding) :- Non-linear dim. Red.

→ Manifold technique. → don't rely on projection

→ Not good for large dataset due to m^2 complex.