10/6/18

Top 5 ML libraries in Python | Udemy

1) Pandas 2) Numpy 3) scikit-learn 4) Matplotlib 5) NLTK

Python is a high level programming language, is case sensitive.
(Human readable)
and easier to work with.

low level : machine language, harder to deal and work with.

libraries: groups of code or some programs that can be imported into Python

Python has become the standard for building applied predictive models.

Model : a group of algos which tend to produce an output.

① Pandas (import pandas as pd)
↳ creating an 'alias'.
(more to deal with data.)

df = pd.read_csv(" ")

df.describe(), df.shape(), ufo.dtypes, ufo.columns, ufo.drop(' ', axis=1, inplace=True), ufo.head(), ufo.drop(['', '', ''], axis=1, inplace=True), ufo.column.sort_values(ascending=False), ufo.sort_values(' '), ufo.sort_values(['', '', ...,''], pd.merge(df1, df2)

Note ufo.head() returns top 5 rows.

Note all methods end with parenthesis, e.g. ufo.head().

A Pandas series is 1D array of indexed data.

② Numpy (for scientific computing in python)

Array: collection of elements of same dtype.

```
n = np.array([1,4,6,6,6])
>> array([1, 4, 6, 6, 6])
type(n) = numpy.ndarray
n.ndim = 1
n.shape = (5,)
n.size = 5
n.dtype = 'int32'
n[0,] = 1   <-- Imp
```

n[6,] : out of bound error.

n[:4] : ([1,2,3,4]) i.e array upto 4 elements.

np.save('n', n) ← n.npy files is saved with n array.

np.load("n.npy") = ([1,2,3,4,5,6])

Note On structural level, an array is basically pointers. It's a combination of a memory address, a data type, a shape and strides.

array of nos. is the most common array.

③ Scikit-Learn (sklearn)

Unlike Numpy and Pandas which are focused on loading, manipulating & summarizing data, scikit is focused on modelling data which provides a range of supervised and unsupervised learning also via a consistent interface in python.

Pythonic way is to import only that part of the library which we need.

e.g. from sklearn import datasets
from " " metrics
from " svm " SVC

```
ds = datasets.load_iris()
model = SVC()                    # classifier to be used on data.
model.fit(ds.data, ds.target)
expected = ds.target
predicted = model.predict(ds.data)
print(metrics.classification_report())   ← (accuracy of model)
                                          └ expected, predicted.
```

» precision recall f1-score support

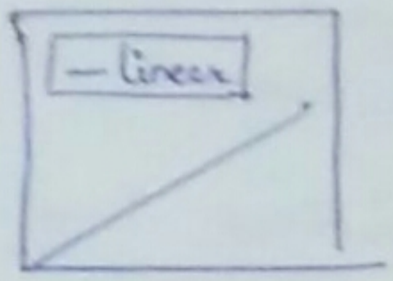
0.99 0.99 0.99 150 ⟶ 99% accuracy

Hence, scikit is focussed on model building

④ Matplotlib (for visualising data)

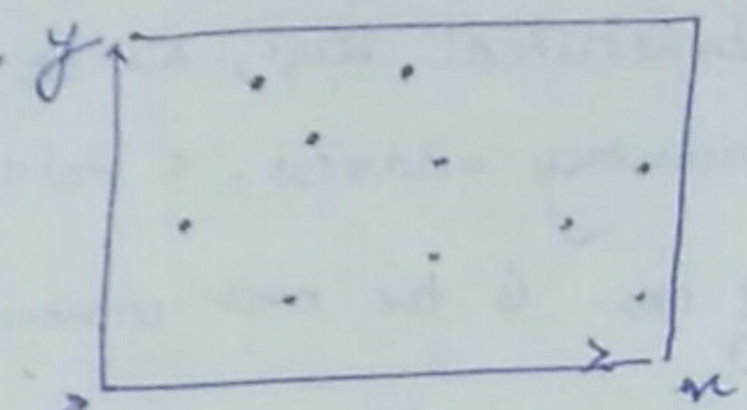
e.g. import matplotlib.pyplot as plt
import numpy as np
x = np.linspace (0, 10, 100)
plt.plot (x, x, label = 'linear')
plt.legend()
plt.show()

»



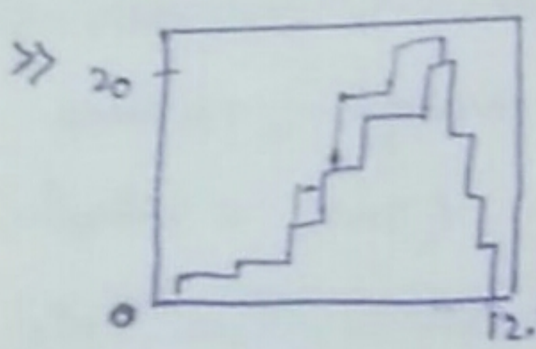
e.g. import matplotlib.pyplot as plt
X = [59, 46 ——, 5]
y = [10, 26, ····, 1]
plt.scatter (x, y)
plt.show()

»



e.g. import numpy as np
import pylab as pl # module in matplotlib that gets installed alongside
ds = np.random. # matplotlib.
normal (5.0, 3.0, 100)

plt.hist (ds)
pl.xlabel ('ds')
pl.show()



(Guassian distribⁿ)

Note 3 most common graphs in matplotlib are 'line', 'scatter' and 'histogram'.

* matplotlib.pyplot is a module that provides an interface that allows to implicitly & automatically create figures and axes to achieve the desired plot.

Figure: overall window or page that everything is drawn on.

Axes: most plotting occurs here. Its an area on which data is plotted and that can have ticks, labels, etc associated with it. Each axes has x and y axis which have major & minor tick lines and tick labels.

Spines: lines that connect the axes tick marks & that designate the boundaries of the data area. They are simple black square that we see when we don't plot any data, but initialized the axes.

⑤ NLTK (Natural language ~~toolkit~~)

e.g. from nltk.tokenize import word_tokenize
word_tokenize ("I like you")
» ['I', 'like', 'you'] # these all are tokens generated.

First kill the stopwords and then count the frequency of the words that are left.  Bg of words.

Tokenize is the process of breaking up a text, not specific to words, sen or para-

eg. from nltk.tokenize import sent_tokenize
sentence = "I love you. You love me".
sen_tokenize (sentence)
» ['I love you.', 'You love me.'] # the sentence can be tokenized into words as well.

So, Tokenize is just the process of breaking up text and two diff. structures.

eg. from nltk.corpus import stopwords.
stopwords.words ("english")
» ['I', 'me', 'myself', 'my' - - -]

Note NLP tasks involve organizing and structuring knowledge.
text in tweets, reviews, news, etc is unstructured data.

⑥ Seaborn (for visualiza of statistical models)
- Seaborn is based on Matplotlib & highly dependent on it. Its visualizations include heat maps, that summarize the data but still depict the overall distributions.

⑦ Gensim (for NLP)

10/6/18

Deep learning libraries in Python

These libraries support a variety of DL architectures such as feed-fwd networks, auto encoders, RNN and CNN.

Libraries

1) Theano (The Mentor)
- low level library that specialises in efficient computa°. This is used directly only when fine grain customiza° & flexibility is needed.

2) Tensorflow (The kid)
- another low level library that is less mature than Theano. It is supported by Google & offers out-of the box distributed computing.

3) Keras (The story)
- heavy wrapper for both Theano & Tensorflow. Its minimalistic, modular and awesome for rapid experimenta°.

① THEANO:

It trades ease of use for flexibility. It is similar to Tensorflow but not that efficient. It helps deal with multi Dimensional arrays which are frustrating with other libraries. It is more of a research tool than production tool.

Pros: i) Flexible ii) Performant if used properly

Cons: i) substantial learning curve
ii) Lower level API
iii) Compiling complex symbolic graphs can be slow.

Theano has led to all DL libraries. It serves as building blocks for NN just like Numpy for scientific computing.
It is hardly used unless in case of lower-level optimization.

② Tensorflow:

Google made it. It allows efficient numerical computation using data flow graphs. Theano is faster than Tensorflow. It provides utilities for efficient data pipelining and has built in models

for inspection, visualize and serialize of models.

Pros: i) Backed by Google.
ii) Very large community
iii) Low level and high level interfaces to network training.
iv) Faster model compilation than Theano-based options.
v) Clean multi GPU support.

Cons: i) Initially slower at many benchmarks than Theano, although it is now catching up.
ii) RNN support still outclassed by Theano.

Primary benefit of Tensorflow over Theano is distributed computing (Hint) Tensorflow is a computational framework for expressing algos involving large no. of Tensors operations.
It is written basically in C and C++ but has a sophisticated frontend for Python. Its flexible architecture allows it to get deployed on 1 or more CPUs or GPUs in a desktop, server or mobile device all with the same API.

③ Keras: Ideal library for rapid experimentation but doesn't support multi GPU environment for parallel training.

It allows to use either Theano or Tensorflow backend, i.e we can switch b/w the two depending on our application.
Keras has out of box implementation of common nw structures. Its fast & easy to get a CNN up and running.

Pros: i) Easier learning curve & provides easier way to express NN.
ii) Intuitive high level interface
iii) Choice of Theano's or Tensorflow's symbolic graph.

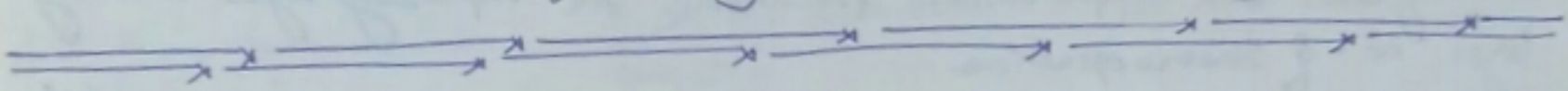
Cons: i) less flexible, more prescriptive than other options.
It helps processing datasets, compiling models, evaluating results, visualize and many more.
ii) Keras is slow compared to other libraries bcoz it constructs a computational graph & then uses it to perform operations.

④ Pyevolve : provides a great framework to build and execute genetic algorithm (basically a search heuristic that mimics the process of natural selection)

③ Keras (contd...)

Keras provides many preprocessed datasets & pretrained models like Incep°, Squeezenet, Mnist, VGG, Resnet, etc.

With Keras, we can easily construct both sequence based networks (where input flows linearly through networks) and graph based networks (where inputs can skip certain layers, only to be concatenated later).



11/6/18

Supervised learning Algos.

- KNN
- Decision Tree & Random Forest
- Naive Bayes
- Boosting and Adaboost.

① KNN :

- Training set includes classes.
- Examine K items near item to be classified.
- New item placed in class with the most no. of close items.
- O(q) for each tuple to be classified. (q is size of training set)

Hence, in KNN, there is no model other than storing the entire dataset. So, there is no learning required. It takes predictions using training dataset directly.

It summarizes the output variable for the most similar K instances (the neighbours) and in regression, this might be the mean or median o/p variable and in classification, this might be the mode (or most common) class value.

The most common distance measure b/w new input & training dataset is Euclidean distance.

↳ Euclidean distance $(x, x_i) = \text{sqrt}(\text{sum}((x_j - x_{ij})^2))$

Other popular distance measures:

- Hamming distance : distance b/w binary vectors.
- Manhattan distance : distance b/w real vectors using sum of their absolute difference. Also called City Block distance.
- Minkowski distance : Generalize of Euclidean & Manhattan.

Note Euclidean is good when all i/p variables are similar in type (e.g. all measured width and height). Manhattan is good if i/p variables are not similar in type (such as age, gender, height.)

computational complexity of KNN increases with increasing size of training dataset.

- diff disciplines of KNN:

① Instance based learning - raw training instances are used to make predictions. As such KNN is referred as instance " or case based learning (where each training instance is a case from the problem domain.)

② Lazy learning - No learning of the model is required and all the work happens when predic° is requested.

③ Non-parametric - KNN makes no assumption about functional form of the problem being solved.

* in binary classifica° problem:

$$p(class=0) = \frac{count(class=0)}{count(class=0) + count(class=1)}$$

Note In case we have even no. of classes, it's better to choose K value with an odd no. to avoid a tie and if classes are odd, choose K to be even, or simply expand K by 1.

Curse of dimensionality: KNN works well with small no. of i/p variables but struggles when no. of i/p is very large.

As no. of dimensions (features) of i/p ↑es, volume of i/p space ↑es. This is bcoz in high Dimension, all points are far from each other and our intuition of distance in 2D or 3D spaces breaks down.

- Data preparing for KNN:

i) Rescale data : Normalize in the range [0,1]. It is a good idea to standarize your data if it has Guassian distribu°.

ii) Address missing data : Remove those i/p with missing features as their distance can't be calculated.

iii) lower dimensionality : KNN works well for lower dimensional data. So, reduce the dimensionality before using KNN.

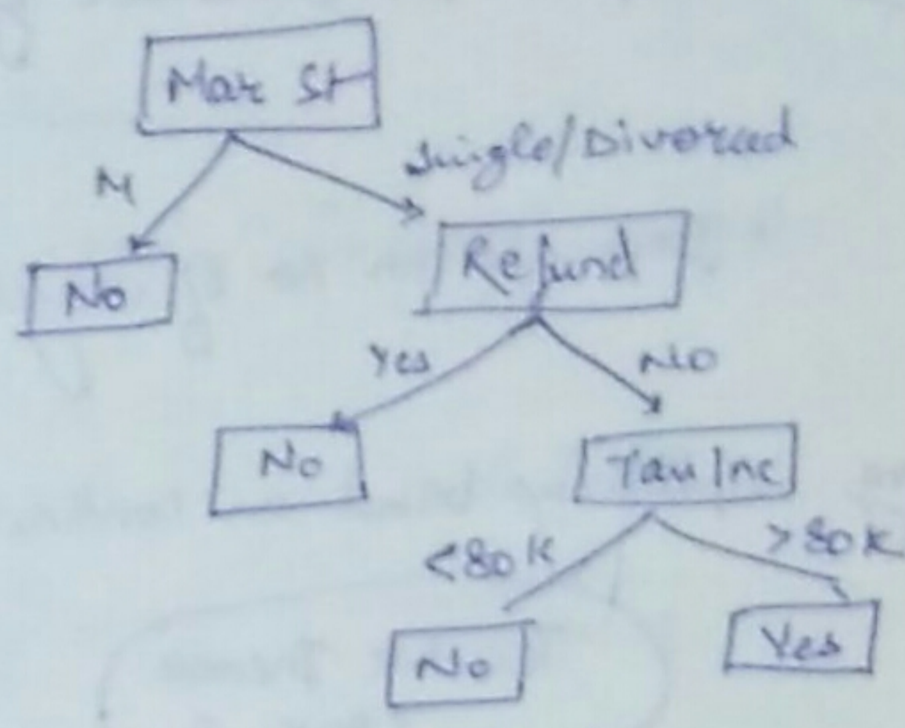
② Decision Tree and Random Forest :

e.g.

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	S	70K	No
4	Yes	M	120K	No
5	No	Divorced	95K	Yes
6	No	M	60K	No
7	Yes	D	220K	No
8	No	S	85K	Yes

Training Data

Model : Decision Tree



There could be more than 1 tree that fits the same data. This decision tree is used to find sol^n to test data.

→ Decision Tree Induc^ Algorithm

- Hunt's algorithm :

i) Let D_t be the set of training records that reach a node t.
ii) If D_t contains records that belong to the same class y_t, then t is a leaf node labeled as y_t.
iii) If D_t is an empty set, then t is a leaf node labeled by default class y_d.
iv) If D_t contains records that belong to more than 1 class, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset.

→ Tree Induction:

- Greedy strategy - split the records based on an attribute test that optimizes certain criterion..

- Issues - i) Determine how to split the records.
  - How to specify attribute test condition?
  - How to determine best split?

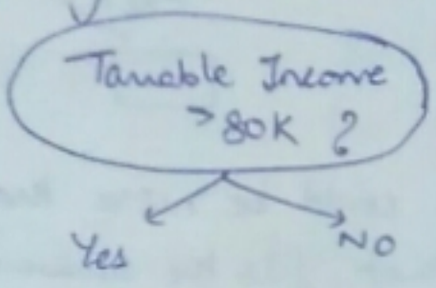
  ii) Determine when to stop splitting.

Q. How to specify test condition?

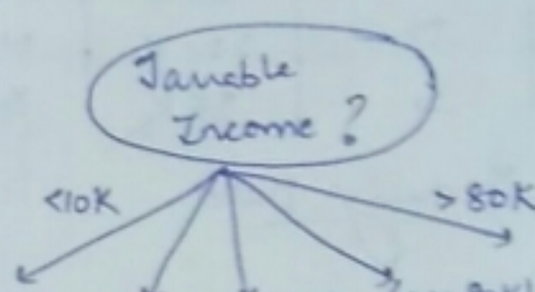
Ans • Depends on attribute types ← Nominal, Ordinal, Continuous.

• Depends on no. of ways to split ← 2-way split (Bin
Multi-way split

eg. Splitting based on continuous attribute:



(i) Binary split



(ii) Mutli way split

Q How to determine the best split?

Ans i) Greedy approach

- Nodes with homogeneous class distribution are preferr
- Needs a measure of node impurity:

C0: 5
C1: 5

Nonhomogeneous, High degree of impurity.

C0: 9
C1: 1

Homogeneous, low degree of impurity

* Measures of Node Impurity ← Gini Index, Entropy, Misclassification Error

① Gini Index - Gini Index for a given node t:

$$GINI(t) = 1 - \sum_j [p(j|t)]^2$$

$p(j|t)$ = relative frequency of class j at node t

Maximum $(1 - \frac{1}{n_c})$ where n_c = no. of classes, when records are equally distributed among all classes, implying least interesting information.

Minimum (0.0) when all records belong to 1 class, implying most interesting information.

e.g.

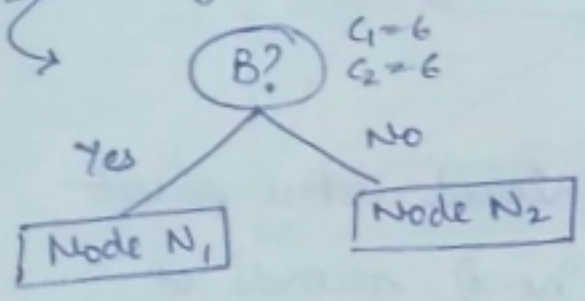
C_1 0	$C_1 = 1$	$C_1 = 2$	$C_1 = 3$
C_2 6	$C_2 = 5$	$C_2 = 4$	$C_2 = 3$
Gini = 0	$Gini = 1 - \left(\left(\frac{1}{6}\right)^2 + \left(\frac{5}{6}\right)^2\right)$ $= 0.278$	Gini = 0.444	Gini = 0.5

When a node p is split into k partitions (children), the quality of split is computed as,

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

where, n_i = no. of records at child i
n = no. of records at node p.

e.g. Larger and purer partitions are sort for.



	N_1	N_2
C_1	5	1
C_2	2	4
Gini = 0.333		

$Gini(N_1) = 1 - (5/6)^2 - (1/6)^2$
$= 0.194$

$Gini(N_2) = 1 - (1/6)^2 - (4/6)^2$
$= 0.528$

$Gini(B) = 1 - (6/12)^2 - (6/12)^2$
$= 0.5$

bcoz $Gini_{split} = \frac{7}{12}(0.194) + \frac{5}{12}(0.528)$
$= 0.333$ (quality of split when B divides into N_1 & N_2)

Chose the split with best Gini Index split
↓
means least value for Gini Index.

② Entropy - entropy at a given node t :

$$\text{Entropy}(t) = -\sum_j p(j|t) \log p(j|t)$$

$p(j|t)$ is relative frequency of class j at node t.

→ Entropy measures homogeneity of node.

- maximum ($\log n_c$) when records are equally distributed among all class implying least info.
- minimum (0.0) when all records belong to 1 class, implying most information.

So, entropy based computation are similar to Gini Index computation.

e.g. | C_1 | 0 |
| C_2 | 6 |

$P(C_1) = 0/6$ $\quad P(C_2) = 6/6$ $\quad \therefore$ Entropy $= -0 \log 0 - 1 \log 1 = 0$

| C_1 | 5 |
| C_2 | 1 |

$P(C_1) = 5/6$ $\quad P(C_2) = 1/6$ $\quad \therefore$ Entropy $= -\frac{1}{6} \log \frac{1}{6} - \frac{5}{6} \log \frac{5}{6}$

$$\text{GAIN}_{\text{split}} = \text{Entropy}(p) - \left| \sum_{i=1}^{k} \frac{n_i}{n} \text{Entropy}(i) \right|$$

↳ Information gain

So, GAIN split measures reduction in entropy achieved when parent node p is split into k partitions, n_i being the no. of records at partition i. Choose the split with maximum GAIN split i.e the split that achieves most reduction.

Disadvantage : Tends to prefer splits that result in large no. of partitions, each being small but pure.

$$\text{GAIN Ratio}_{\text{split}} = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

n_i = no. of records in partition i.

↳ to overcome the above disadvantage.

③ Misclassification Error – classification error at node t:

$$\text{Error}(t) = 1 - \max P(i|t)$$

- maximum $(1 - 1/n_c)$ when records are equally distributed among all classes.
- minimum (0.0) when all records belong to 1 class, implying most interesting info.

e.g.

C_1	0
C_2	6

$P(C_1) = 0/6 = 0$ $\quad P(C_2) = 6/6 = 1$

$\text{Error} = 1 - \max(0, 1) = 0$

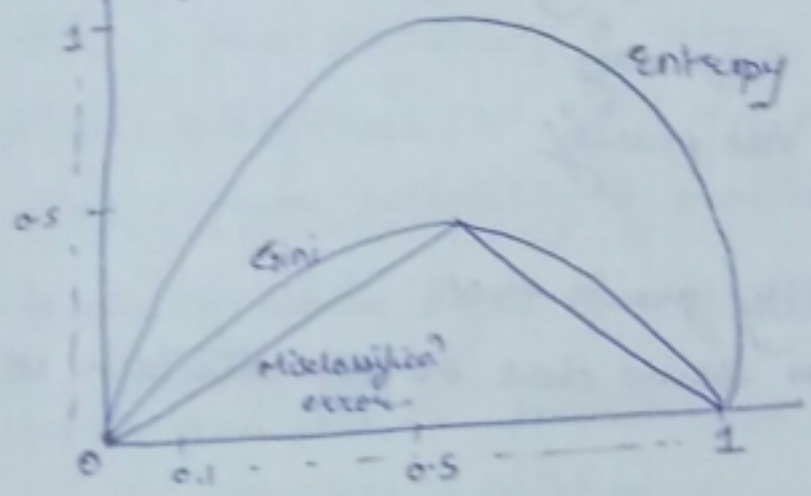
C_1	1
C_2	5

$P(C_1) = 1/6$ $\quad P(C_2) = 5/6$

$\text{Error} = 1 - \max(1/6, 5/6) = 1/6$

} chose the one with minimum error.

for a binary (2-class) problem:



CART :
(classification & Regression Tree used for building both classification & regression decision trees)

Q. When to Stop splitting ?

Ans - Stop expanding when all the records belong to the same class.
- " " a node when all the records have similar attribute values.
- Early termination.

Q. Advantages of Decision Tree ?

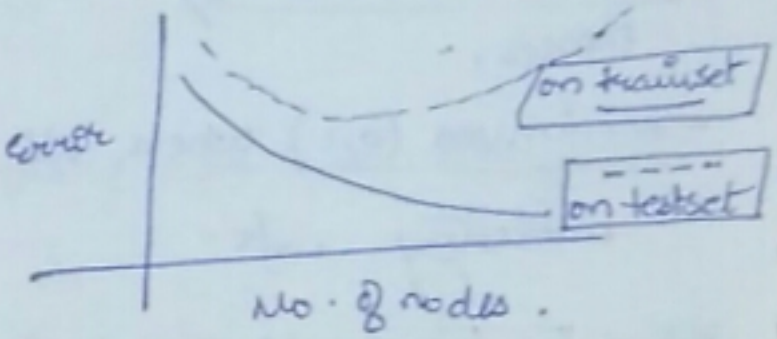
i) Inexpensive to construct.
ii) Extremely fast at classifying unknown records.

iii) easy to interpret for small sized trees.
iv) Comparable accuracy with other classifica techniques for many simple datasets.

Q. Practical issues of classification?

Ans i) Underfitting and Overfitting
ii) Missing values
iii) Costs of classification

Overfitting



when model is too simple, underfit happens with both train & test errors high.

Note Given two models with same error, one should prefer simpler model over the more complex model, bcoz there is a greater chance that complex model was fitted accidentally by errors in data.

Q. How to stop overfitting? < Prepruning / Post pruning

Ans ① Prepruning
- Stop before it becomes a fully grown tree.
- Stop if all instances belong to same class or all attribute values are same.
- Stop if no. of instances is less than some user-specified threshold.
- Stop if class distribu of instances are independent of available features. (e.g. using χ^2 test)
- Stop if expanding the current node doesn't improve impurity measures (e.g. Gini or information gain)

② Postpruning
- Grow decision tree to its full and trim the nodes in a bottom-up fashion. If error improves after trimming, replace subtree by leaf node and its label is determined from majority class label in subtree.

(3) Random Forest : 12/6/18

Refer to blue copy notebook (spiral) named 'Paperisto'.

(4) Naive Bayes : extremely fast relative to other classifica° algos. based on Bayes probability theorem to predict the class of unknown dataset. It is easy to build and particularly useful for very large datasets. It is simple but outperforms many sophisticated classifica° algos.

It assumes that presence of particular feature in a class is unrelated to the presence of any other feature, so is 'naive'.

$$P(c|x) = \frac{P(x|c)\,P(c)}{P(x)}$$

{ P(c) and P(c|x) can be calculated directly from the given dataset. }

$$P(c|x) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

P(c|x) : probability of class c (target) given x (predictors).

P(c) : prior probability of class.

P(x|c) : likelihood which is the probability of predictor given class.

P(x) : prior probability of predictor.

Refer to e.g. of Naive Bayes from blue spiral notebook ('Paperisto')

Pros : i) easy and fast to predict class of test data, also performs well in multi class prediction.

ii) mostly used in text classifica° & with problems having multiple classes.

iii) when assumption of independence holds, NB performs better than LR and we need less training data.

iv) performs well in case of categorical i/p variables compared to numerical variables. for numerical variable, normal distribu° is assumed (bell curve, which is a strong assump°)

Cons : i) If categorical variable has a category in test dataset, which was not observed during training, then model will

assign a zero probability and will be unable to make a prediction, often known as 'zero frequency'. for this we can use the laplace estima^n, smoothing technique.

ii) NB is also known as a bad estimator, so probabity o/p are not to be taken too seriously.

iii) NB assumes predictors as independent, but in real life it is impossible to have completely independent predictors.

→ Applica^ of NB

1) Real time predic^ : bcoz it is fast
2) Multi class predic^ : good at it
3) Text classifica^ / spam filtering / Sentiment analysis : bcoz NB is good multi class targets and indepent variables.
4) Recommendation System : NB + Collabrative filtering build a good recommender.

Naive Bayes Models

Guessian	Multinomial	Bernoulli
- it assumes that features follow a normal distribu^. - It is used in classifica^.	- used for discrete counts. - in text classifica^. - checks the no. of times outcome number x_i is observed over the n trials.	- useful when feature vectors are binary. - BOW is one applica^ with '0' meaning words doesn't occur in the document and '1' meaning, it does.

Q How to improve NB?

Ans i) Convert features to normal distribution using transforma^, etc.
ii) If test data has 'zero frequency' issue, use smoothing techniques.
iii) Remove correlated features bcoz otherwise they would be voted twice in the model leading to over inflating importance.

iii) As NB has less options for parameter tuning, focus more on data preprocessing and feature selection.

v) NB has no variance to minimize and so no ensembling methods help.

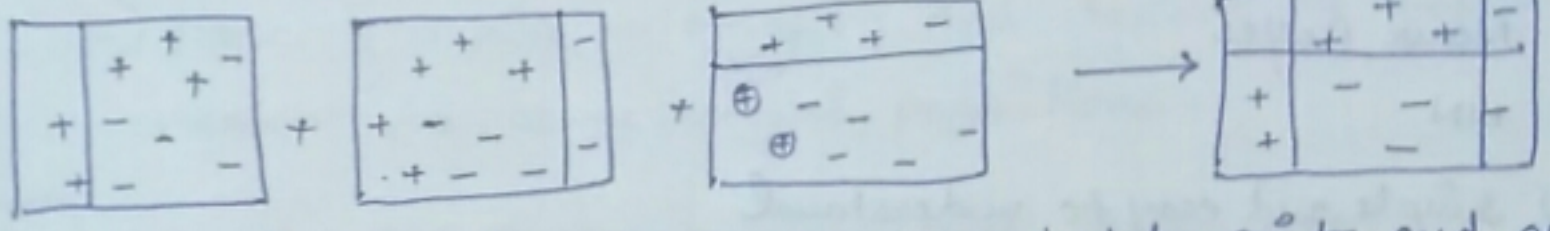
(5) Boosting and Adaboost :

Boosting refers to family of algos which convert weak learners to strong learners. To do this, it combines the predic^n of each weak learner by:

- using avg. or weighted avg.
- considering predic^n as higher vote.

Boosting algos < Adaboost (Adaptive boosting)
Gradient Tree Boosting.

① Adaboost :



What we did is, assign equal weights to each data point and applied a decision boundary to classify them as + or −. Then the ones wrongly classified are given more weightage and another decision boundary is applied. Hence in end we get a complex rule as compared to individual weak learners. Repeat until higher accuracy is achieved. Any ML algo can be used as base learner if it accepts weight on training dataset. Adaboost can be used for both classifica^n & regression.

② Gradient Boosting : {GD: Gradient Descent}

It trains many model sequentially and each new model gradually minimizes the loss f^N ($y = ax+b+e$, e needs special atten^n as it is the error term) of the whole system using GD method. The learning model constructively fit new models to provide a more accurate estimate of target.

Note Parametric and Non Parametric ML algos.

① Parametric : Assump° can greatly simplify the learning process, but can also limit learning. Algos that simplify the fN to a known form are called parametric ML algos.

The algo involves two steps < select a form for the fN
learn coeff. from training data for the fN.

Now matter how much data is thrown at a parametric model, it won't change its mind about how many parameters it need.

Often the assumed functional form is a linear combin° of i/p variable and so parametric ML algos are often also called 'Linear ML algos'.

e.g.s. Logistic Regression
Perceptron
Naive Bayes.
NN

Pros- i) simple and easy to understand
ii) fast to learn from data
iii) require less training data & can work well even if the fit to data is not perfect.

Cons - i) constrained to the chosen form and its parameters.
ii) limited complexity and so more suited to simpler problems.
iii) They have a poor fit and unlikely to match the actual underlying mapping function.

② Non parametric : Algos that do not make strong assump° about the form of mapping fN. They are thus free to learn any functional form from the training data.

These are good in trxe of lot of data & no prior knowledge and

when we don't worry much about choosing just the right features. The non-parametric methods seek to best fit the training data in constructing the mapping fn while maintaining some ability to generalize on unseen data. As such, they can fit a large no. of functional forms.

e.g. KNN
Decision Trees
SVM

Pros - i) flexibility to fit a large no. of functional forms.
ii) No assumpⁿ about underlying data
iii) Can result in higher performance models for prediction.

Cons - i) Require a lot more training dataset.
ii) a lot slower to train as they have far more parameters.
iii) More of a risk to overfit and harder to explain the reason for some specific predictions.

Note Low bias ML algos : Decision Tree, KNN, SVM
High bias " " : Linear Regression, LDA, logistic Regression
low variance " " : " " " "
High variance " " : Decision Tree, KNN, SVM

Apart from bias & Variance error, there is irreducible error. It can't be reduced regardless of the algo used. It is the error introduced from the chosen framing of the problem and may be caused by unknown variables that influence the mapping of i/p variables to o/p variables.

Q What can you do if your model outputs -ve values in a case where -ve response is not possible?

Ans. i) Convert -ve responses to zero.

ii) Take log of the response variable and then fit the data. After that take antilog of the predicted variable to get actual answer in +ve range.

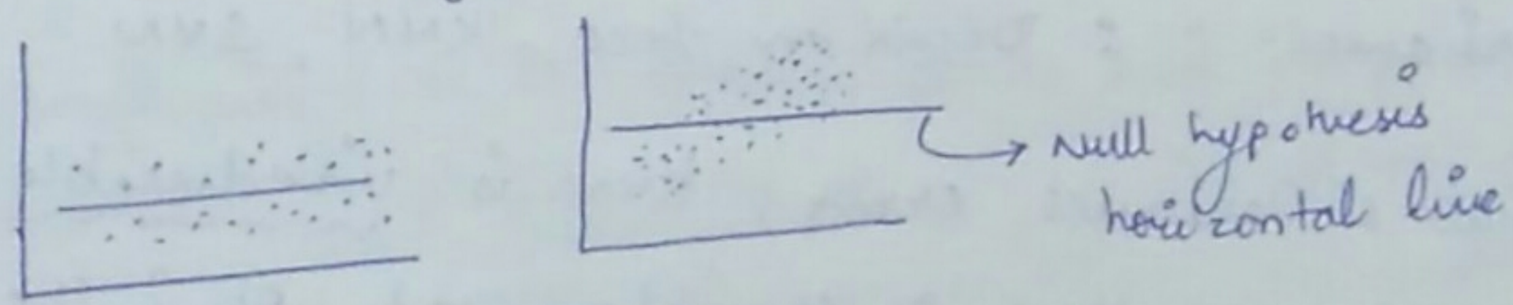
Q When is R^2 -ve?

Ans R^2 compares fit of chosen model with that of a horizontal st. line (the null hypothesis). If the chosen model fits worse than a horizontal line, then R^2 is -ve. R^2 is not always the square of anything.

$$R^2 = 1 - \frac{SS_{reg}}{SS_{total}} \quad (\text{when } SS_{reg} > SS_{total},\ R^2 \text{ is -ve})$$

With linear regression with no constraints, R^2 must be +ve (or zero) and equals the square of the correla coefficient, r. A-ve R^2 is only possible c linear regression with either the intercept or the slope are constrained so that the "best fit" line (given the constraint) fits worse than a horizontal line. With non-linear regression, R^2 can be -ve whenever the best fit model fits the data worse than a horizontal line.

→ This horizontal line goes through the mean of all Y values.



$$R^2 = 1 - \frac{SS_{reg}}{SS_{Total}}$$

$$= 1 - \frac{\sum_{i=1}^{n} e_i^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

e_i → distance from regression line

$(y_i - \bar{y})$ → distance from horizontal line