

[Open in app](#)[Sign up](#)[Sign In](#)

Source: Pixabay

You have **2 free member-only stories left** this month. [Sign up](#) for Medium and get an extra one.

◆ Member-only story

Pros and cons of various Machine Learning algorithms



Shailaja Gupta · Follow

Published in Towards Data Science

8 min read · Feb 28, 2020

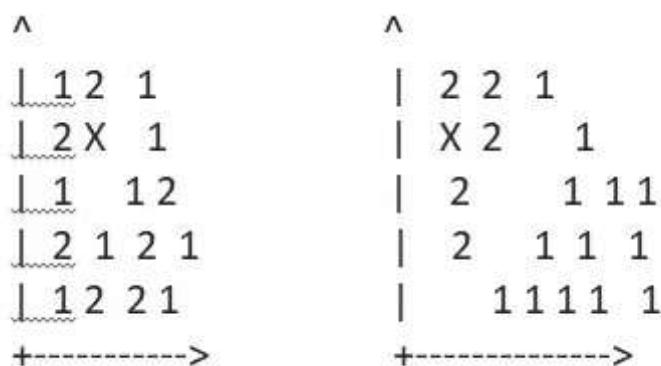
[Listen](#)[Share](#)

There are many classification algorithms in machine learning. But ever wondered which algorithm should be used for what purpose and what kind of application. If yes, then please read the pros and cons of various machine learning algorithms used in classification. I have also listed down their use cases and applications.

SVM (Support Vector Machine)

Pros

1. **Performs well in Higher dimension.** In real world there are infinite dimensions (and not just 2D and 3D). For instance image data, gene data, medical data etc. has higher dimensions and SVM is useful in that. Basically when the number of features/columns are higher, SVM does well
 2. **Best algorithm when classes are separable.** (when instances of both the classes can be easily separated either by a straight line or non-linearly). To depict separable classes, lets take an example(here taking an example of linear separation, classes can also be non-linearly separable, by drawing a parabola for e.g. etc). In first graph you cannot tell easily whether X will be in class 1 or 2, but in case 2 you can easily tell that X is in class 2. Hence in second case classes are linearly separable.



First is non-separable class, second is separable class.

- 3. **Outliers** have less impact.
 - 4. SVM is suited for extreme case binary classification.

Cons:

1. **Slow:** For larger dataset, it requires a large amount of time to process.
 2. **Poor performance with Overlapped classes :** Does not perform well in case of overlapped classes.
 3. **Selecting appropriate hyperparameters is important:** That will allow for sufficient generalization performance.

4. Selecting the appropriate kernel function can be tricky.

Applications:

Bag of words application(many features and columns), speech recognition data, classification of images(non-linear data), medical analytics(non linear data), text classification(many features)

Naive Bayes

Pros

1. **Real time predictions:** It is very fast and can be used in real time.
2. **Scalable with Large datasets**
3. **Insensitive to irrelevant features.**
4. **Multi class prediction** is effectively done in Naive Bayes
5. **Good performance with high dimensional data**(no. of features is large)

Cons

1. **Independence of features does not hold:** The fundamental Naive Bayes assumption is that each feature makes an independent and equal contribution to the outcome. However this condition is not met most of the times.
2. **Bad estimator:** Probability outputs from predict_proba are not to be taken too seriously.
3. **Training data should represent population well:** If you have no occurrences of a class label and a certain attribute value together (e.g. class="No", shape="Overcast") then the posterior probability will be zero. So if the training data is not representative of the population, Naive bayes does not work well.(This problem is removed by smoothening techniques).

$$P(x | c) = P(\text{Sunny} | \text{Yes}) = 3 / 9 = 0.33$$

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3

		Play Golf		
		Yes	No	
Outlook	Sunny	3/9	2/5	5/14
	Overcast	4/9	0/5	4/14
	Rainy	2/9	3/5	5/14

$$\begin{aligned} P(x) &= P(\text{Sunny}) \\ &= 5 / 14 = 0.36 \end{aligned}$$

$$P(c) = P(\text{Yes}) = 9 / 14 = 0.64$$

Posterior Probability:

$$P(c | x) = P(\text{Yes} | \text{Sunny}) = 0.33 \times 0.64 / 0.36 = 0.60$$

Posterior Probability

$$P(\text{No} | \text{Overcast}) = P(\text{Overcast} | \text{No}) \times P(\text{No}) / P(\text{Overcast}) = (0 / 5) \times (5 / 14) / (4 / 14) = 0$$

Applications:

Naive Bayes is used in Text classification/ Spam Filtering/ Sentiment Analysis. It is used in text classification (it can predict on multiple classes and doesn't mind dealing with irrelevant features), Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative sentiments), recommendation systems (what will the user buy next)

Logistic Regression

Pros

1. Simple to implement
2. Effective
3. Feature scaling not needed: Does not require input features to be scaled (can work with scaled features too, but doesn't require scaling)
3. Tuning of hyperparameters not needed.

Cons

1. Poor performance on non-linear data(image data for e.g)

2. Poor performance with irrelevant and highly correlated features (use Boruta plot for removing similar or correlated features and irrelevant features).
3. Not very powerful algorithm and can be easily outperformed by other algorithms.
4. High reliance on proper presentation of data. All the important variables / features should be identified for it to work well.

Applications:

Any classification problem that is preferably binary (it can also perform multi class classification, but binary is preferred). For example you can use it if your output class has 2 outcomes; cancer detection problems, whether a student will pass/fail, default/no default in case of customer taking loan, whether a customer will churn or not, email is spam or not etc.

Random Forest

Pros:

1. Random forest can **decorrelate trees**. It picks the training sample and gives each tree a subset of the features(suppose training data was [1,2,3,4,5,6], so one tree will get subset of training data [1,2,3,2,6,6]. Note that size of training data remains same, both datas have length 6 and that feature '2' and feature '6' are repeated in the randomly sampled training data given to one tree. Each tree predicts according to the features it has. In this case tree 1 only has access to features 1,2,3 and 6 so it can predict based on these features. Some other tree will have access to features 1,4,5 say so it will predict according to those features. If features are highly correlated then that problem can be tackled in random forest.
2. **Reduced error:** Random forest is an ensemble of decision trees. For predicting the outcome of a particular row, random forest takes inputs from all the trees and then predicts the outcome. This ensures that the individual errors of trees are minimized and overall variance and error is reduced.

3. **Good Performance on Imbalanced datasets :** It can also handle errors in imbalanced data (one class is majority and other class is minority)
4. **Handling of huge amount of data:** It can handle huge amount of data with higher dimensionality of variables.
5. **Good handling of missing data:** It can handle missing data very well. So if there is large amount of missing data in your model, it will give good results.
6. **Little impact of outliers:** As the final outcome is taken by consulting many decision trees so certain data points which are outliers will not have a very big impact on Random Forest.
7. **No problem of overfitting:** In Random forest considers only a subset of features, and the final outcome depends on all the trees. So there is more generalization and less overfitting.
8. **Useful to extract feature importance** (we can use it for feature selection)

Cons:

1. Features need to have **some predictive power** else they won't work.
2. Predictions of the trees need to be uncorrelated.
3. **Appears as Black Box:** It is tough to know what is happening. You can at best try different parameters and random seeds to change the outcomes and performance.

Applications:

Credit card default, fraud customer/not, easy to identify patient's disease or not, recommendation system for ecommerce sites.

Decision Trees

Pros

1. **Normalization or scaling of data not needed.**

2. Handling missing values: No considerable impact of missing values.

3. Easy to explain to non-technical team members.

4. Easy visualization

5. Automatic Feature selection : Irrelevant features won't affect decision trees.

Cons

1. Prone to overfitting.

2. Sensitive to data. If data changes slightly, the outcomes can change to a very large extent.

3. Higher time required to train decision trees.

Applications:

Identifying buyers for products, prediction of likelihood of default, which strategy can maximize profit, finding strategy for cost minimization, which features are most important to attract and retain customers (is it the frequency of shopping, is it the frequent discounts, is it the product mix etc), fault diagnosis in machines(keep measuring pressure, vibrations and other measures and predict before a fault occurs) etc.

XGBoost

Pros

1. Less feature engineering required (No need for scaling, normalizing data, can also handle missing values well)

2. Feature importance can be found out(it output importance of each feature, can be used for feature selection)

3. Fast to interpret

4. Outliers have minimal impact.

5. Handles large sized datasets well.

6. Good Execution speed

7. Good model performance (wins most of the Kaggle competitions)

8. Less prone to overfitting

Cons

1. Difficult interpretation , visualization tough

2. Overfitting possible if parameters not tuned properly.

3. Harder to tune as there are too many hyperparameters.

Applications

Any classification problem. Specially useful if you have too many features and too large datasets, outliers are present, there are many missing values and you don't want to do much feature engineering. It wins almost all competitions so this is an algo you must definitely have in mind while solving any classification problem.

k-NN (K Nearest Neighbors)

Pros

1. Simple to understand and implement

2. No assumption about data (for e.g. in case of linear regression we assume dependent variable and independent variables are linearly related, in Naïve Bayes we assume features are independent of each other etc., but k-NN makes no assumptions about data)

3. Constantly evolving model: When it is exposed to new data, it changes to accommodate the new data points.

4. Multi-class problems can also be solved.

5. One Hyper Parameter: K-NN might take some time while selecting the first hyper parameter but after that rest of the parameters are aligned to it.

Cons

1. Slow for large datasets.
2. **Curse of dimensionality:** Does not work very well on datasets with large number of features.
3. Scaling of data absolute must.
4. **Does not work well on Imbalanced data.** So before using k-NN either undersample majority class or oversample minority class and have a balanced dataset.
5. Sensitive to outliers.
6. Can't deal well with missing values

Applications:

You can use it for any classification problem when dataset is smaller, and has lesser number of features so that computation time taken by k-NN is less. If you do not know the shape of the data and the way output and inputs are related (whether classes can be separated by a line or ellipse or parabola etc.), then you can use k-NN.

Conclusion

As the models came later , they became more and more advanced. So usually if you look at a model strictly from the performance perspective, then neural networks, XGBoost etc. are the best models since they are relatively recent. However , different models work better for different data.

For e.g. if features are highly independent then Naive Bayes will work great. If there are too many features and dataset is medium sized, SVM is good. If there is linear

relationship between dependent and independent variables, then linear regression, logistic regression, SVM are good. If dataset is small and you don't know the relationship between the dependent and independent variable, then you can use k-NN. Hence before you decide on which ML algorithm to use, you must know and analyse the data. If you are unable to zero in on one machine learning algorithm, then you can evaluate all the models and check their accuracies on training and test set and then finalize one model.

[Ai And Data Science](#)[Machine Learning](#)[Classification](#)[Algorithms](#)[Data Science](#)[Follow](#)

Written by Shailaja Gupta

140 Followers · Writer for Towards Data Science

IIM Indore alumnus, Machine Learning and Data Science enthusiast

More from Shailaja Gupta and Towards Data Science



 Shailaja Gupta in Towards Data Science

Data Science for Beginners

What all do we need to enter Data Science?

• 7 min read • Jan 28, 2020

 33 





 Jacob Marks, Ph.D. in Towards Data Science

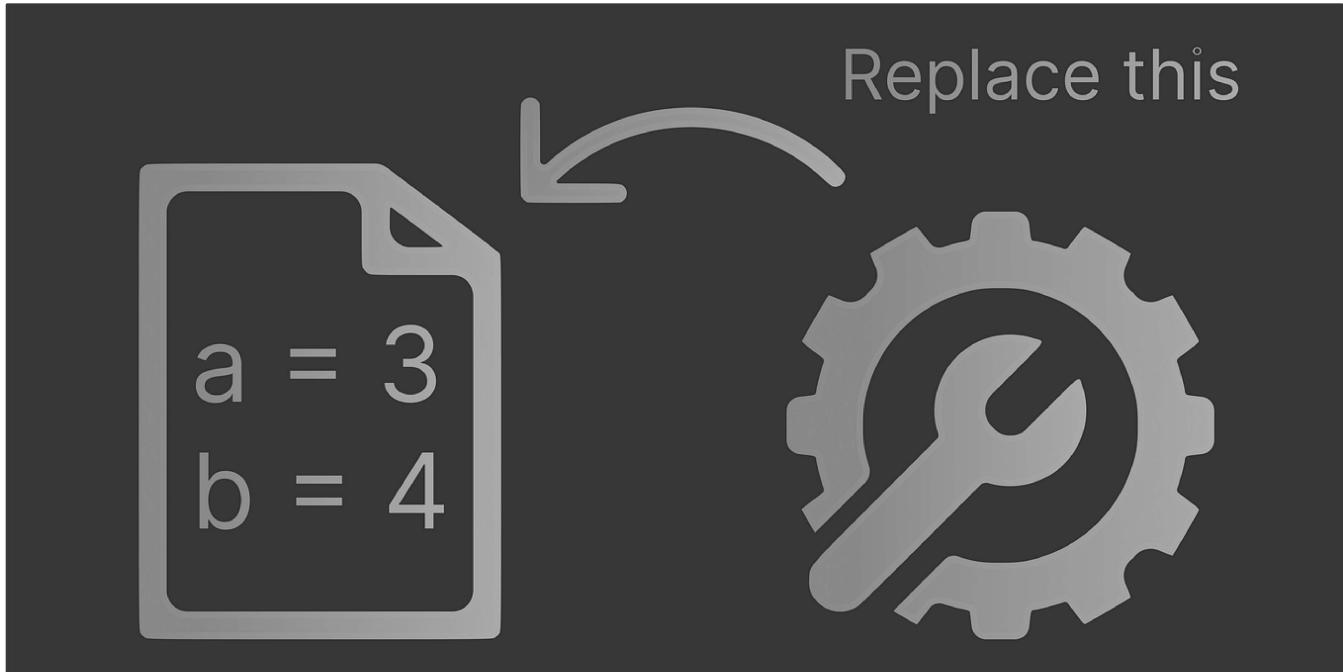
How I Turned My Company's Docs into a Searchable Database with OpenAI

And how you can do the same with your docs

15 min read · Apr 25

👏 4.1K

💬 49



👤 Khuyen Tran in Towards Data Science

Stop Hard Coding in a Data Science Project—Use Config Files Instead

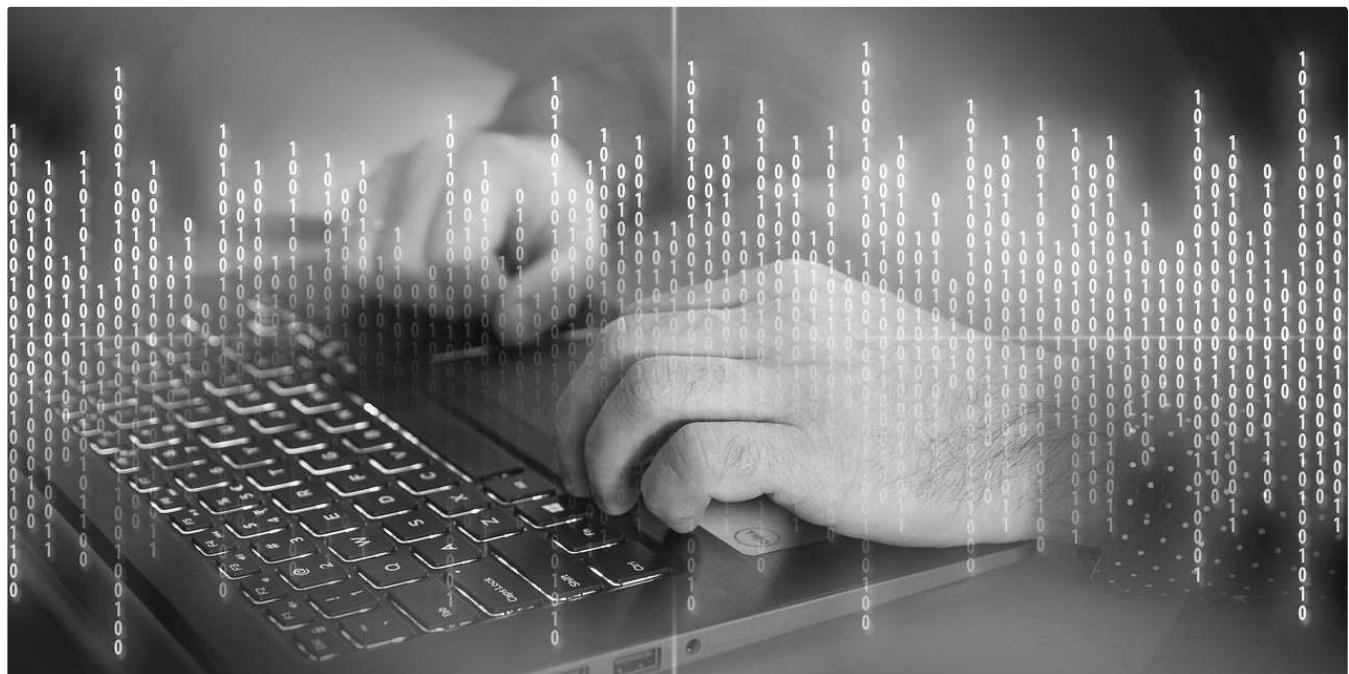
And How to Efficiently Interact with Config Files in Python

⭐ · 6 min read · May 26

👏 1.6K

💬 20





Shailaja Gupta in Towards Data Science

How to tackle any classification problem end to end & choose the right classification ML algorithm.

We are aware of many machine learning algorithms now. However when I was learning ML , I was unable to find an article covering all ML...

◆ · 14 min read · Feb 28, 2020

102

3



See all from Shailaja Gupta

See all from Towards Data Science

Recommended from Medium



Matt Chapman in Towards Data Science

The Portfolio that Got Me a Data Scientist Job

Spoiler alert: It was surprisingly easy (and free) to make

• 10 min read • Mar 24

👏 3.1K

💬 46



Samuel Flender in Towards Data Science

Class Imbalance in Machine Learning Problems: A Practical Guide

Five lessons from the trenches of applied data science

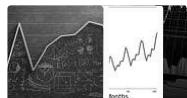
◆ · 8 min read · Oct 3, 2022

👏 313

💬 2



Lists



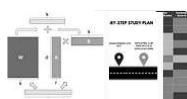
Predictive Modeling w/ Python

18 stories · 12 saves



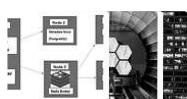
Practical Guides to Machine Learning

10 stories · 25 saves



Natural Language Processing

346 stories · 7 saves



New_Reading_List

173 stories · 1 save



✍ Amy @GrabNGoInfo in GrabNGoInfo

Top 7 Support Vector Machine (SVM) Interview Questions for Data Science and Machine Learning

Margin, soft margin, support vectors, C, Gamma, Kernel trick, hinge loss, and pros and cons of a support vector machine (SVM) model

◆ · 5 min read · Feb 9

👏 1



+



 Zach Quinn in Pipeline: Your Data Engineering Resource

3 Data Science Projects That Got Me 12 Interviews. And 1 That Got Me in Trouble.

3 work samples that got my foot in the door, and 1 that almost got me tossed out.

◆ · 7 min read · Aug 29, 2022

👏 3.2K

💬 46

+



Albers Uzila in Level Up Coding

Wanna Break into Data Science in 2023? Think Twice!

It won't be smooth sailing for you

• 11 min read • Dec 23, 2022

906

16



Samuel Flender in Towards Data Science

How I Cracked the Meta Machine Learning Engineering Interview

Practical tips for the coding, design, and behavior rounds

• 7 min read • Oct 25, 2022

963

3



See more recommendations