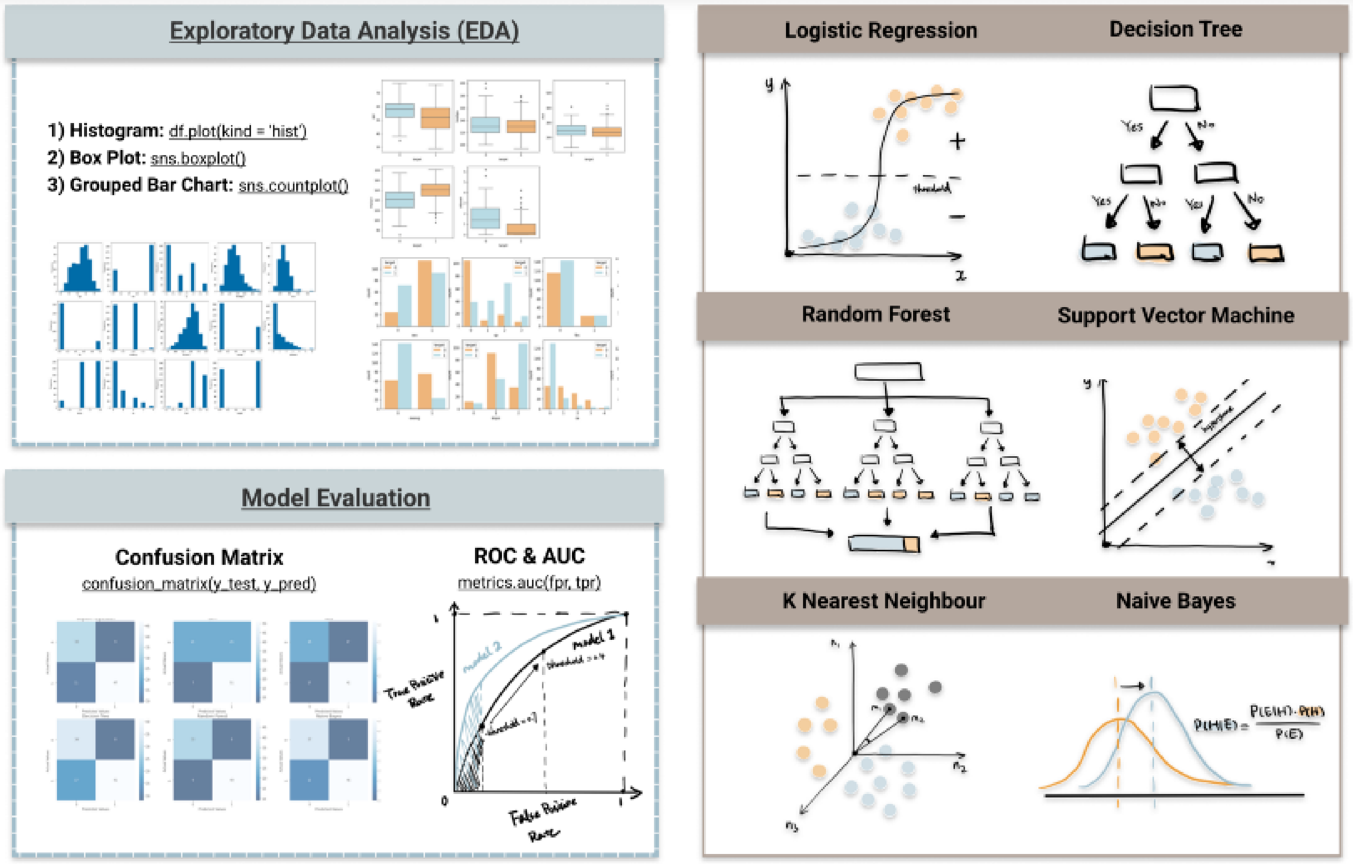
Destin Gong Follow

Feb 22 · 9 min read · · Listen

Top 6 Machine Learning Algorithms for Classification

How to Build a Machine Learning Model Pipeline in Python



Machine Learning Algorithms for Classification (original image from my website)

Supervised vs. Unsupervised vs. Reinforcement Learning

The easiest way to distinguish a supervised learning and unsupervised learning is to see whether the data is labelled or not.

Supervised learning learns a function to make prediction of a defined label based on the input data. It can be either classifying data into a category (classification problem) or forecasting an outcome (regression algorithms).

Unsupervised learning reveals the underlying pattern in the dataset that are not explicitly presented, which can discover the similarity of data points (clustering algorithms) or uncover hidden relationships of variables (association rule algorithms) …

Reinforcement learning is another type of machine learning, where the agents learn

Classification vs Regression

Supervised learning can be furthered categorized into classification and regression algorithms. **Classification model** identifies which category an object belongs to whereas **regression model** predicts a continuous output.

For a guide to regression algorithms, please see:

Top 4 Regression Algorithms in Machine Learning

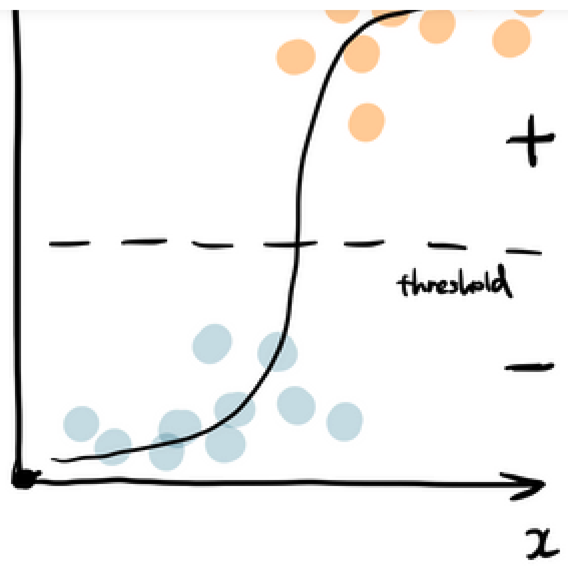
A Comprehensive Guide to Implementation and Comparison

towardsdatascience.com

Sometimes there is an ambiguous line between classification algorithms and regression algorithms. Many algorithms can be used for both classification and regression, and classification is just regression model with a threshold applied. When the number is higher than the threshold it is classified as true while lower classified as false.

In this article, we will discuss top 6 machine learning algorithms for classification problems, including: l*ogistic regression, decision tree, random forest, support vector machine, k nearest neighbour and naive bayes*. I summarized the theory behind each as well as how to implement each using python. Check out the code for model pipeline on my website.

1. Logistic Regression



logistic regression (image by author)

Logistics regression uses sigmoid function above to return the probability of a label. It is widely used when the classification problem is binary — true or false, win or lose, positive or negative …

The sigmoid function generates a probability output. By comparing the probability with a pre-defined threshold, the object is assigned to a label accordingly. Check out my posts on logistic regression for a detailed walkthrough.

Simple Logistic Regression in Python

Step-by-Step Guide from Data Preprocessing to Model Evaluation

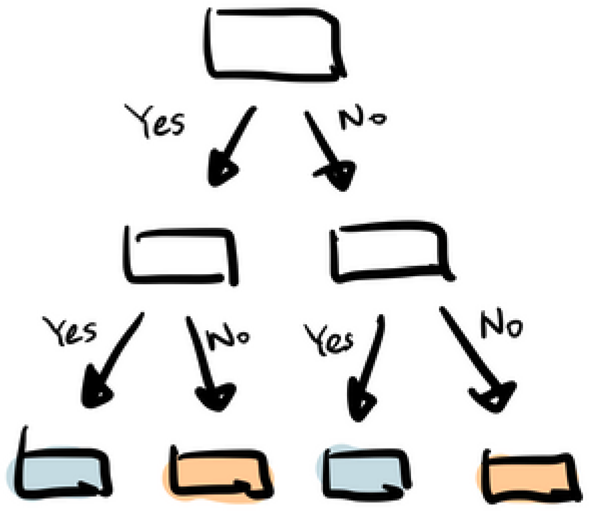
towardsdatascience.com

Below is the code snippet for a default logistic regression and the common hyperparameters to experiment on — see which combinations bring the best result.

```
  from sklearn.linear_model import LogisticRegression
```

logistic regression common hyperparameters: penalty, max_iter, C, solver

2. Decision Tree

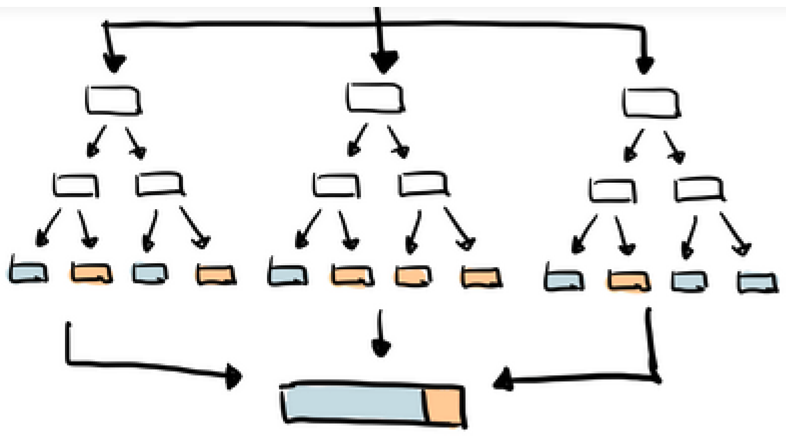


decision tree (image by author)

Decision tree builds tree branches in a hierarchy approach and each branch can be considered as an if-else statement. The branches develop by partitioning the dataset into subsets based on most important features. Final classification happens at the leaves of the decision tree.

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
y_pred = dtc.predict(X_test)
```

decision tree common hyperparameters: criterion, max_depth, min_samples_split, min_samples_leaf; max_features



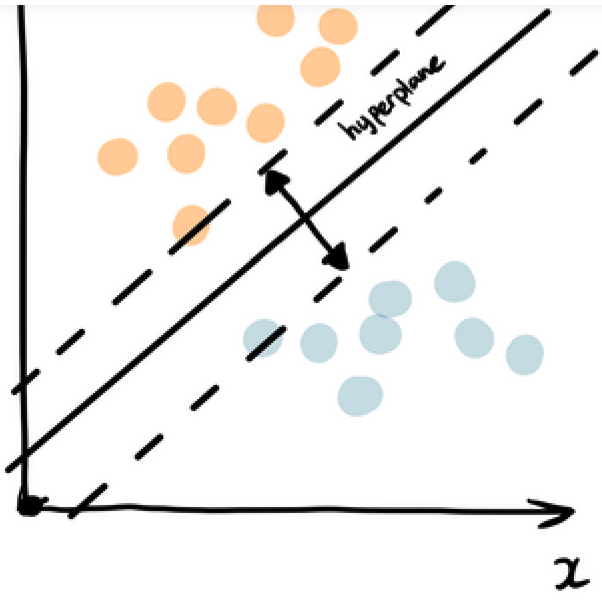
random forest (image by author)

As the name suggest, random forest is a collection of decision trees. It is a common type of ensemble methods which aggregate results from multiple predictors. Random forest additionally utilizes bagging technique that allows each tree trained on a random sampling of original dataset and takes the majority vote from trees. Compared to decision tree, it has better generalization but less interpretable, because of more layers added to the model.

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
y_pred = rfc.predict(X_test)
```

random forest common hyperparameters: n_estimators, max_features, max_depth, min_samples_split, min_samples_leaf, boostrap

4. Support Vector Machine (SVM)



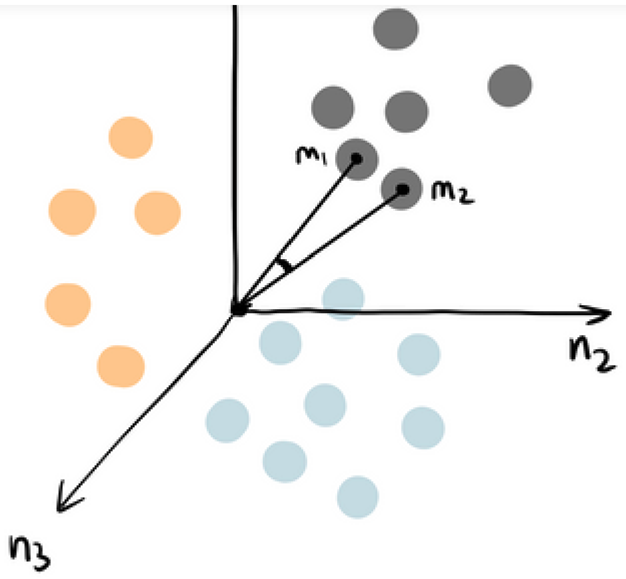
support vector machine (image by author)

Support vector machine finds the best way to classify the data based on the position in relation to a border between positive class and negative class. This border is known as the hyperplane which maximize the distance between data points from different classes. Similar to decision tree and random forest, support vector machine can be used in both classification and regression, SVC (support vector classifier) is for classification problem.

```
from sklearn.svm import SVC
svc = SVC()
svc.fit(X_train, y_train)
y_pred = svc.predict(X_test)
```

support vector machine common hyperparameters: c, kernel, gamma

5. K-Nearest Neighbour (KNN)



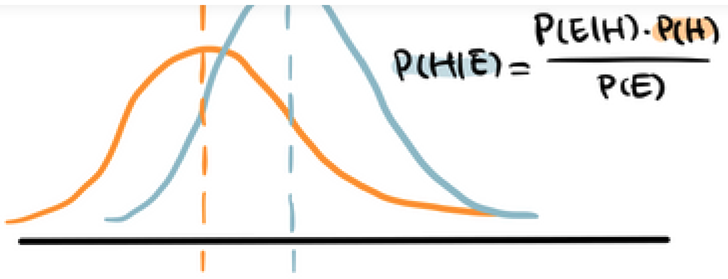
knn (image by author)

You can think of k nearest neighbour algorithm as representing each data point in a n dimensional space — which is defined by n features. And it calculates the distance between one point to another, then assign the label of unobserved data based on the labels of nearest observed data points. KNN can also be used for building recommendation system, check out my article on "Collaborative Filtering for Movie Recommendation" if you are interested in this topic.

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

KNN common hyperparameters: n_neighbors, weights, leaf_size, p

6. Naive Bayes



naive bayes (image by author)

Naive Bayes is based on Bayes' Theorem — an approach to calculate conditional probability based on prior knowledge, and the naive assumption that each feature is independent to each other. The biggest advantage of Naive Bayes is that, while most machine learning algorithms rely on large amount of training data, it performs relatively well even when the training data size is small. Gaussian Naive Bayes is a type of Naive Bayes classifier that follows the normal distribution.

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)
```

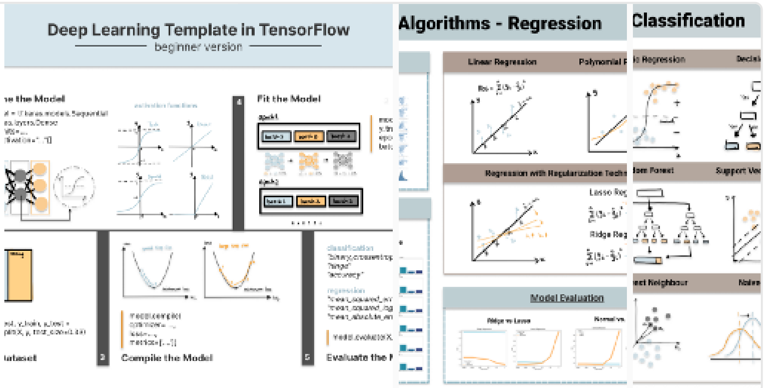
gaussian naive bayes common hyperparameters: priors, var_smoothing

If you want to know more other machine learning algorithms, check out my list:

Destin Gong

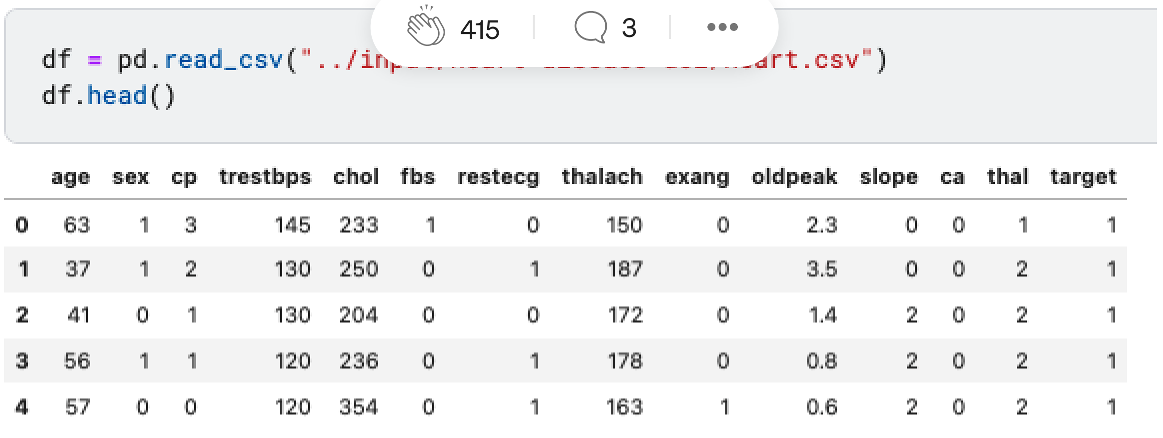
Practical Guides to Machine Learning

View list 8 stories



1. Loading Dataset and Data Overview

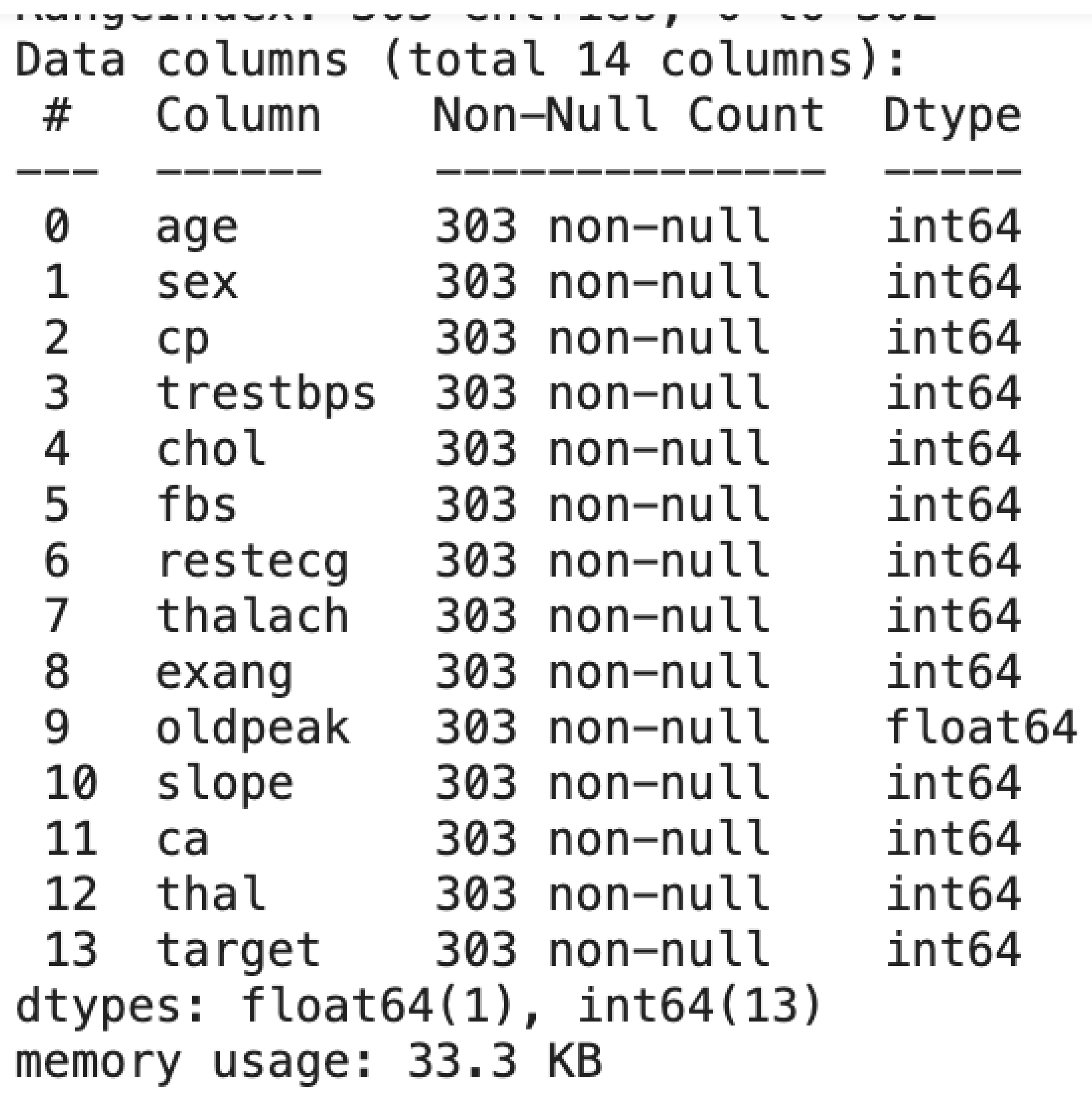
I chose the popular dataset Heart Disease UCI on Kaggle for predicting the presence of heart disease based on several health related factors.



```
df = pd.read_csv("../in                          art.csv")
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Use `df.info()` to have a summarized view of dataset, including **data type, missing data and number of records.**



```
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

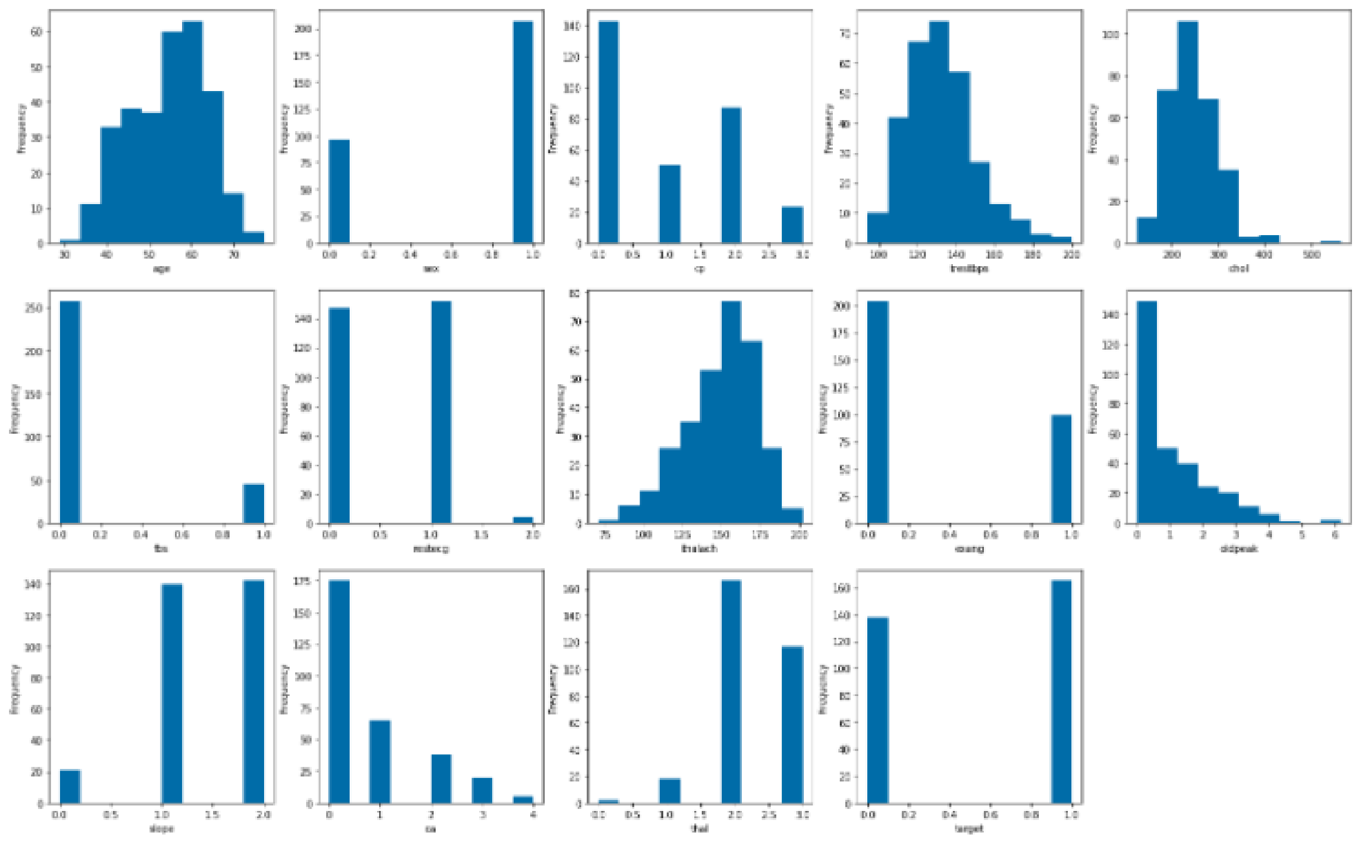
2. Exploratory Data Analysis (EDA)

Histogram, grouped bar chart and box plot are suitable EDA techniques for classification machine learning algorithms. If you'd like a more comprehensive guide to EDA, please see my post "Semi-Automated Exploratory Data Analysis Process in Python"

Semi-Automated Exploratory Data Analysis Process in Python

This article covers several techniques to automate the EDA process using Python, including univariate analysis...

```
fig = plt.figure(figsize=(24, 15))
i = 0
for column in df:
    sub = fig.add_subplot(3, 5, i + 1)
    sub.set_xlabel(column)
    df[column].plot(kind = 'hist')
    i = i + 1
```

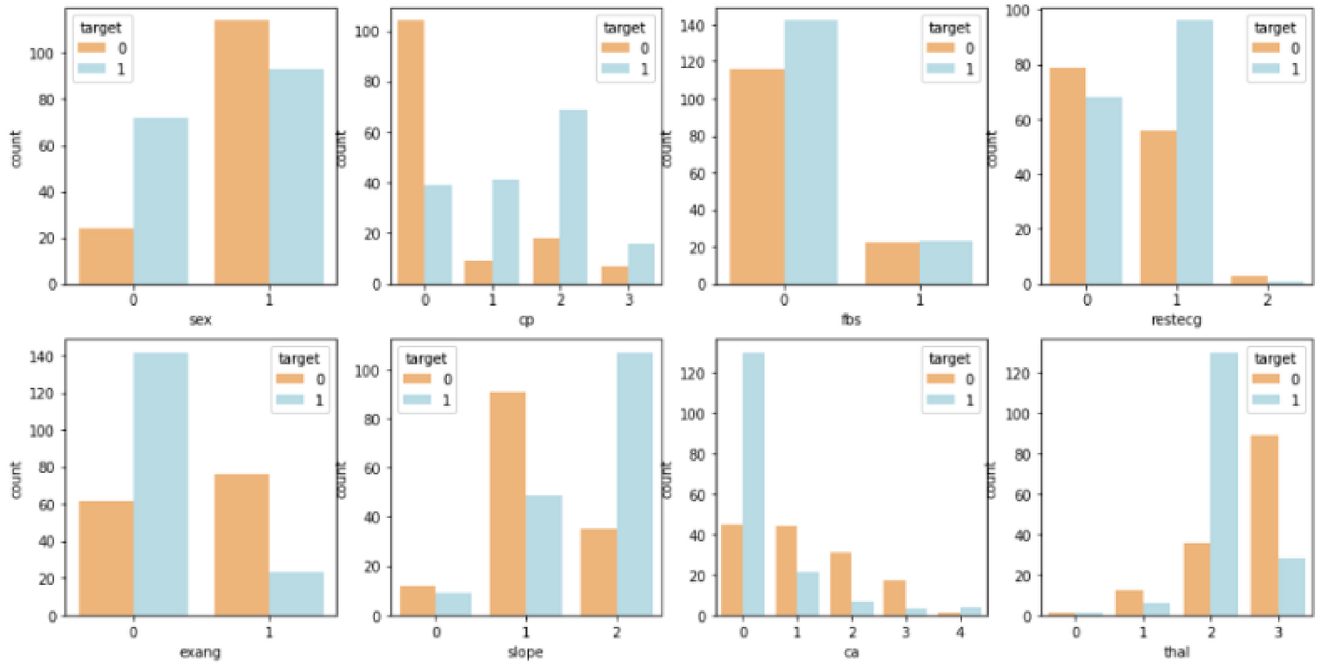


univariate analysis (image by author)

Histogram is used for all features, because all features have been encoded into numeric values in the dataset. This saves us the time for categorical encoding that usually happens during the feature engineering stage.

Categorical Features vs. Target — Grouped Bar Chart

```
for i in range(len(cat_list)):
    column = cat_list[i]
    sub = fig.add_subplot(2, 4, i + 1)
    chart = sns.countplot(data = df,x= column, hue= 'target', palette = 'RdYlBu')
```

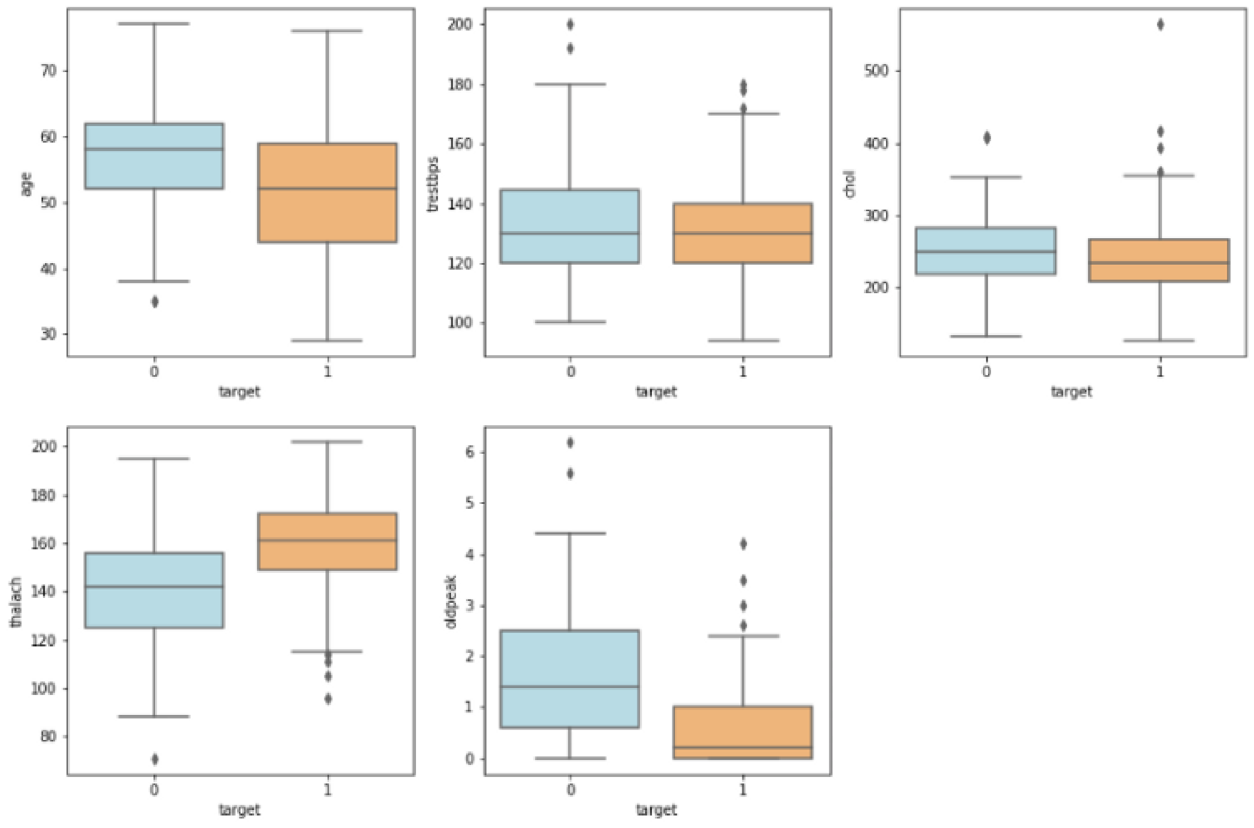


grouped bar chart (image by author)

To show how categorical value weigh in determining the target value, grouped bar chart is a straightforward representation. For example, sex = 1 and sex = 0 have distinctly distribution of target value, which indicates it is likely to contribute more to the prediction of target. Contrarily, if the target distribution is the same regardless of the categorical features, then very likely they are not correlated.

Numerical Features vs. Target — Box Plot

```
for i in range(len(num_list)):
    column = num_list[i]
    sub = fig.add_subplot(2, 3, i + 1)
    sns.boxplot( x = 'target', y = column, data = df, palette = "RdYlBu_r")
```



box plot (image by author)

Box plot shows how the values of numerical features varies across target groups. For example, we can tell that “oldpeak” have distinct difference when target is 0 vs. target is 1, suggesting that it is an important predictor. However, ‘trestbps’ and ‘chol’ appear to be less outstanding, as the box plot distribution is similar between target groups.

3. Split Dataset into Training and Testing Set

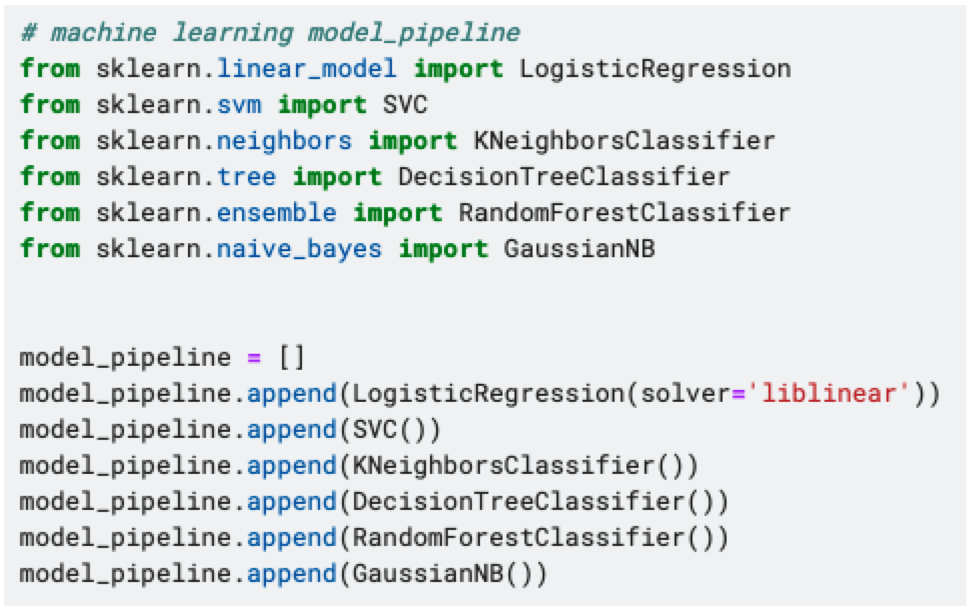
Classification algorithm falls under the category of supervised learning, so dataset needs to be split into a subset for training and a subset for testing (sometime also a validation set). The model is trained on the training set and then examined using the testing set.

```
X = df.drop(['target'], axis=1)
y = df["target"]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)
```

4. Machine Learning Model Pipeline

In order to create a pipeline, I append the default state of all classification algorithms mentioned above into the model list and then iterate through them to train, test, predict and evaluate.

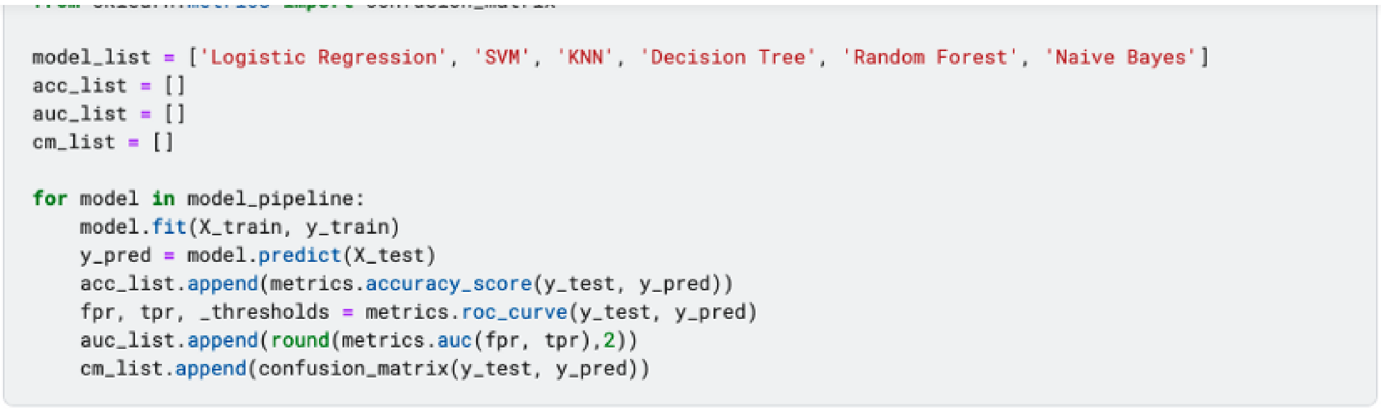


```
# machine learning model_pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB


model_pipeline = []
model_pipeline.append(LogisticRegression(solver='liblinear'))
model_pipeline.append(SVC())
model_pipeline.append(KNeighborsClassifier())
model_pipeline.append(DecisionTreeClassifier())
model_pipeline.append(RandomForestClassifier())
model_pipeline.append(GaussianNB())
```

model pipeline (image by author)

5. Model Evaluation



```
model_list = ['Logistic Regression', 'SVM', 'KNN', 'Decision Tree', 'Random Forest', 'Naive Bayes']
acc_list = []
auc_list = []
cm_list = []

for model in model_pipeline:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc_list.append(metrics.accuracy_score(y_test, y_pred))
    fpr, tpr, _thresholds = metrics.roc_curve(y_test, y_pred)
    auc_list.append(round(metrics.auc(fpr, tpr),2))
    cm_list.append(confusion_matrix(y_test, y_pred))
```

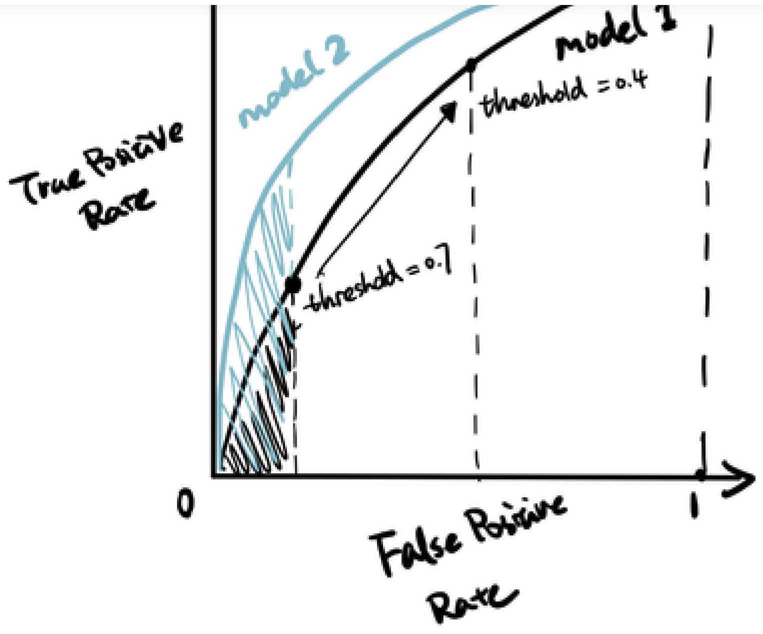
model evaluation (image by author)

Below is an abstraction explanation of commonly used evaluation methods for classification models — **accuracy, ROC & AUC and confusion matrix**. Each of the following metrics is worth diving deeper, feel free to visit my article on logistic regression for a more detailed illustration.

1. Accuracy

Accuracy is the most straightforward indicator of the model performance. It measure the percentage of accurate predictions: *accuracy = (true positive + true negative) / (true positive + false positive + false negative + false positive)*

2. ROC & AUC

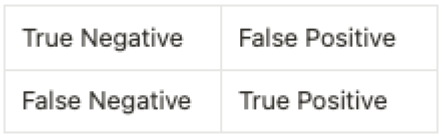


ROC & AUC (image by author)

ROC is the plot of **true positive rate against false positive rate** at various classification threshold. AUC is the area under the ROC curve, and higher AUC indicates better model performance.

3. Confusion matrix

Confusion matrix indicates the actual values vs. predicted values and summarize the **true negative, false positive, false negative and true positive values** in a matrix format.

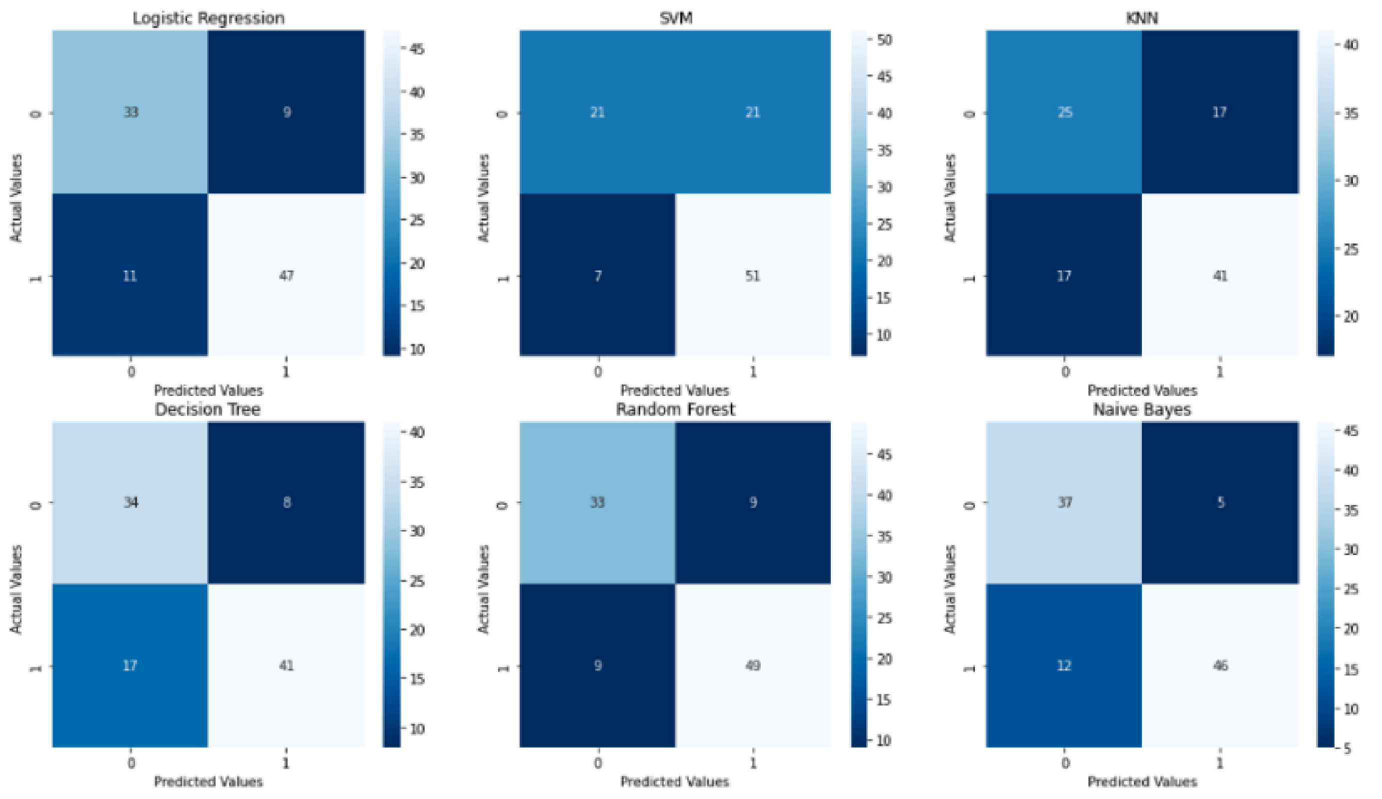


True Negative	False Positive
False Negative	True Positive

Then we can use seaborn to visualize the confusion matrix in a heatmap.

Get unlimited access Open in app

```
for i in range(len(cm_list)):
    cm = cm_list[i]
    model = model_list[i]
    sub = fig.add_subplot(2, 3, i+1).set_title(model)
    cm_plot = sns.heatmap(cm, annot=True, cmap = 'Blues_r')
    cm_plot.set_xlabel('Predicted Values')
    cm_plot.set_ylabel('Actual Values ')
```



confusion matrix plot (image by author)

```
## accuracy and AUC
result_df = pd.DataFrame({'Model':model_list, 'Accuracy': acc_list, 'AUC': auc_list})
result_df
```

	Model	Accuracy	AUC
0	Logistic Regression	0.80	0.80
1	SVM	0.72	0.69
2	KNN	0.66	0.65
3	Decision Tree	0.75	0.76
4	Random Forest	0.82	0.82
5	Naive Bayes	0.83	0.84

random forests and naive bayes are superior algorithms. We can only say that they are more suitable for this dataset where the size is relatively smaller and data is not at the same scale.

Each algorithm has its own preference and require different data processing and feature engineering techniques, for example KNN is sensitive to features at difference scale and multicollinearity affects the result of logistic regression. Understanding the characteristics of each allows us to balance the trade-off and select the appropriate model according to the dataset.

Thanks for reaching so far, if you'd like to read more articles from Medium and also support my work, I really appreciate you signing up Medium Membership using this affiliate link.

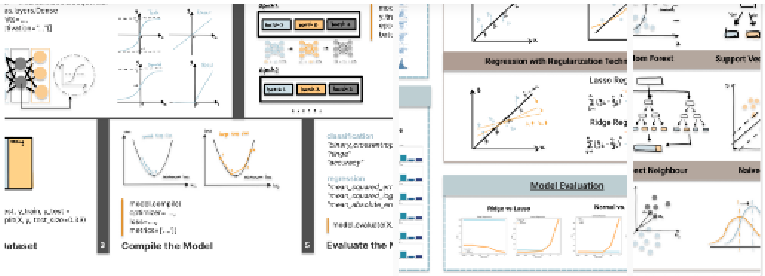
Take Home Message

This article is an introduction of following 6 machine learning algorithms and a guide to build a model pipeline to address classification problems:

1. Logistic Regression
2. Decision Tree
3. Random Forest
4. Support Vector Machine
5. KNN
6. Naive Bayes

Get unlimited access

Open in app



Practical Guides to Machine Learning

View list

8 stories

How to Self-Learn Data Science in 2022

A Project Based Approach to Get Started in Data Science

towardsdatascience.com

Destin Gong

Get Started in Data Science

View list

8 stories