Unsupervised Learning. Day 28.

Association-Rule Technique.

↳ Help to build the ~~recommended~~ system.
recommentation

when we are buying a product what is prob. of buying another similar product.

Ex. Amazon, Netflix, Youtube

- **Purpose:**
To find the frequently occuring combination of itemsets.

- **Support** → Prob. of occuring (buying). → Ex. What is prob. of specific item. buying colgate Brush.

- **confidence**
conditional prob. of occuring combination of itemsets.

Ex. what is prob of buying colgate brush along with colgate paste.

Rules.
What is Association Rule?
study of 'what goes with what' customer who bought x also bought y.
- what symptoms go with what diagnosis.

Methods of Association Rule Technique.

↳ Aprion Algorithm
(Use to find the frequently occuring combination of items.)

↳ MBA (Market Basket Analysis)
(combining fast moving combination product with slow moving combination product).

- Transaction Based or event based
(each every transaction, what is combination of products customer purchased).

Also called 'MBA' and 'affinity analysis'

June 2011

Wk	Mo	Tu	We	Th	Fr	Sa	Su
22			1	2	3	4	5
23	6	7	8	9	10	11	12
24	13	14	15	16	17	18	19
25	20	21	22	23	24	25	26
26	27	28	29	30			

Saturday
May 2011
21

Week 20 ▪ 141-224

Association Rule Mining.

Finding frequent patterns, associations, correlations or casual structures among sets of items in transaction DataBases, relational DB.

Applicn: Basket data analysis, cross-marketing, clustering etc.

Ex: Rule form:

"Antecedent → consequence [support, confidence]"

Diapers → Beers [0.5%, 60%]

Frequent terms used in this Model.

- Items (I): ie product
- Transaction (t): ie set of items ($t \subseteq I$)
- Transaction DataBase (T): ie set of Transactions, $T = \{t_1, t_2, \ldots t_n\}$

Apriori principle:

↳ Any subset of frequent itemset must be frequent.
(used to find frequently occuring combination of ~~datasets~~ itemset).

Ex.

list of Items in Store.

1 → Milk
2 → Jam
3 → Bread
4 → Wheat Bread
5 → Butter

DataBase D.

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

• listing all occurances

How many times item appear in Data-Base

{1} → 2
{2} → 3
{3} → 3, (itemset, sup)
{4} → 1
{5} → 3 (itemset, sup)

Item 4 has least no. of occurance (remove from list).

Sunday
142-223
22

Updated list →	itemset	sup	Apply combinations → itemset	sup.
	{1}	2	{1 2}	1
	{2}	3	{1 3}	2
	{3}	3	{1 5}	1
	{5}	3	{2 3}	2
			{2 5}	3
			{3 5}	2

(Find in DB, how many customer purchased these combination & put it here)

now removing least occured.

Updated:

itemset	sup.
{1,3}	2
{2 3}	2
{2 5}	3
{3 5}	2

Aim: (Find the freq. occuring itemset)

combining

1 3 2 → 1

1 2 3 5 → 1

1 3 5 → 2

2 3 5 → 2

remove least occured.

final combination (frequently occured combination)

{2 3 5} → 2

* Association Rule is applicable to all the sectors.

Statistical significance of Rules,

- Test of proportions
- Formal adjustment of statistical significance. (control prob. of Type I error).

Multilevel Association Rule.

If the customer who buy 1 product, what is prob. of buying another product. If he buys these 2 product, what is prob. of buying 3rd product

(Items at lower level are expected to have lower support)

Market Basket Analysis

↳ combining fast moving product with slow moving products.
(making combo.)

Important aspect:
- Customer Demographics
- Order characteristics
- Item Popularity.
(New Product Intro. strategy).

→ colgate Babrush + colgate Paste + added new product (Vedshakti) 9

if people like it, they will also seperately buy. (after using). 10

But if we introduce new product seperately, people may not buy it so we make & launch with combo. 11 12 1 2

Variations/Extension in Association Rule. 3

- Disassociation Rules 4
- Sequential Analysis using AssociationRules 5
- Association based Clustering. 6

Over a period of time, customer buying prefrences gets changed. Based on this the recommendation also gets changed.

Practical Implementation.

```
import pandas as pd
df = [['Milk','Bread','Wheat Bread'],['Jam','Bread',
'Butter'],['Milk','Jam','Butter','Bread'],
['Jam','Bread']]
```


```
from mlxtend.preprocessing import TransactionEncoder
                                  convert text into Boolean bcz Apriori understand only Boolean.
te = TransactionEncoder()
te_ary = te.fit(df).transform(df)

df1 = pd.DataFrame(te_ary, columns=
                                te.columns_)

print(df1)

from mlxtend.frequent_patterns import apriori
apriori(df1, min_support=0.1)
                    ↳ min. occurance

freq = apriori(df1, min_support=0.1, use_colnames=True)
                                  uses colnames (items) instead of 0,1,2,3..

# to find the occurances length; ie how many times particular item is occured.

freq['len'] = freq['itemset'].apply(lambda x: len(x))
freq
  ↳ print along with length.

Eg: length should be 2. (50% occurance.)   2/4 total items
freq[(freq['len']==2) & (freq['support']>=0.5)]

Based on this you can frame diff. cond^n.
```

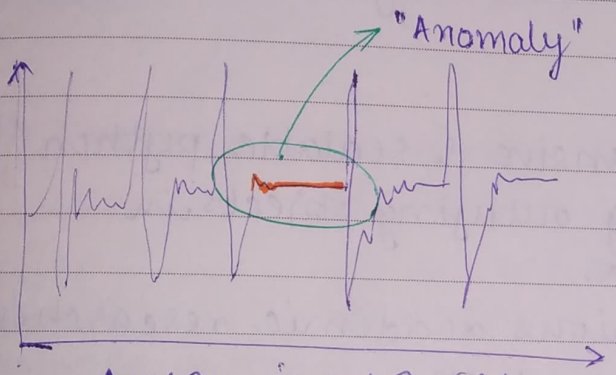
Project Building Session. (Project 1.)

Anomaly Detection in Machine Learning.

it can be said as outliers.

something abnormal (uncertainity / Noise / Disturbance).

Ex =



Here is a pattern in all, but the green circle denotes that there is some abnormal pattern in it.

- Anomaly detection is a technique used to identify unusual patterns that donot confirm to expected behaviours, called outliers.
- It is also called Novelty Detection, Outlier Detection, Forgery Detection, & Out-of-distribution Detection.
- Anomalies are datapts. that are inconsistent with distribution of majority of datapoints.

Novelty Detection
↳ if you have any new features which is similar to object features.

Anomaly Detection
↳ different features which has been included as actual input.

Types (Area/Domain) of Anomaly Detection.

→ Time series anomaly detection
→ Video-level anomaly detection.
→ Image-level
  1. Anomaly classification target
  2. out-of-distribution (OOD) Detection target
  3. Anomaly Segmentation target

PyOD

Python Outlier Detection tool.

- PyOD is a comprehensive & scalable python toolkit for detecting outlying objects in multivariate data.
- PyOD is used in various academic researches & commercial products.
- optimized performance with JIT & parallelization when possible. using numba & joblib

packages used to deploy any model

Practical Implementation,

- install library → pip install pyod.
  or use
  conda install -c conda-forge pyod

Introduction. (1) Linear Models for outlier Detection

↳ PCA (Principal component Analysis)
↳ use sum of weighted projected dist. to eigenvector hyperplane as outlier scores.

Ex PCA: → identify contribution of every individual feature. Based on contribution can we remove any feature.

A	B	C	D	E	F (target)
40	35	22	1	2	

used to find significant features out of entire population.

significant features.

can be removed as they are not much affecting our target var.

- MCD (minimum covariance determinant).

(When minimum covariance is there, then there is no outlier)

differences b/w the std wrt all its value.

- OCSVM (One class support vector Machine)

(Hyperplane concept of SVM).

We put all inline datapoint in one class & outlier in other class

② Proximity Based outlier detection.

closeness

Sunday 29
149-216

Ex. Proximity sensors used in cars / parking system
if your car is going close to other car, alarm shouts !!!

- LOF (Local Outlier Factor)

if you have one set of data usual pts. in which you have so close outlier it is called LOF

- CBLOF (Clustering Based LOF)

if you have 2 classes, behind these you have any other category, it is outlier.

- KNN (~~K neigh~~ K-nearest neighbors).
  ↳ (use distance of kth nearest neighbor as outliers).

- HBOS (Histogram Based outlier score)

(3) Probabilistic Models for outlier Detection.
- ABOD (Angle Based Outlier detection)

(4) Outlier ensembles & combination frameworks
optimization.
- Isolation forest
- Feature Bagging.

Practical Implementation → check file provided by LMS.

- Import Python Packages, Metrics Packages, Pyod Package
  :

```
from scipy.io import loadmat
```
matfiles are matlab files

- Define data file & read X & y.
  Here X & y are defined in matfiles.
  matfiles are different from csv, excel files. They are in dict. format.

```
mat['y'].ravel()
```
converting 2D to 1D.

- contamination
  ↓
  Based on what factors algo. is going to be detect outlier.

```
clf.decision_function(X_test)
```
similar to predict fn in ML.

Here we are using Pyod

8 Wednesday

June 2011

Week 23 ▪ 159-206

Day 29 → K Means Clustering.

(Implementation from scratch.)

Build step by step every step with Python.

— refer Notebook (.ipynb)

Day 30. →

- Practical implementation on Attrition dataset.

```
col_list = ['Age', 'Education', 'TotalWorkingYears',
            'YearsInCurrentRole']

df = data[data.columns[data.columns.isin(col_list)]
df.head()
```

(taking only 4 cols randomly.)

Import kmeans algo.

```
from sklearn.cluster import KMeans
clust = KMeans(n_clusters = 3)
clust.fit(df)
```

→ randomly taken as 3. (we dnt know how many clusters are there).

To estimate clustering performance we have metrix →

```
from sklearn.metrics
        import silhouette_score
labels = clust.labels_
silhouette_score(df, labels)
```

→ 0.3532 (o/p).

method which compute performance score of every data point with respect to cluster.

It lies in b/w −1 to +1.

↓ worst fit. (poor clustering)

↳ very dense clustering.

If you have 0 that means datapoints are overlapped.

ex (same datapt is in cluster A & cluster B also.)

range	Interpretation
0.71 - 1.0	A strong structure has been found
0.51 - 0.7	A reasonable structure " "
0.26 - 0.5	The structure is weak & could be artificial
< 0.25	No substantial structure is found.

- Ploting Elbow curve to get optimum k-value.

```
k = list(range(1, 11))
wcss = []
for i in k:
    kmeans = KMeans(n_clusters = i, random_state = 1,
                    init = 'k-means++')
    kmeans.fit(df)
    labels = kmeans.labels_
    wcss.append(kmeans.inertia_)
plt.plot(k, wcss, color='m')
plt.xlabel('Number of clusters')
plt.ylabel('within cluster sum of sq. errors')
plt.show()
```

default value in kmeans.

↳ By plot we got k = 2,

** Ploting Bar Graph.

```
cluster = kMeans(n_clusters = 2, random_state = 1)
df['cluster'] = cluster.fit_predict(df)
df.cluster.value_counts().sort_index().plot(kind='bar')
plt.xlabel('Cluster Number')
plt.ylabel('Size')
plt.show()
```

10 Friday June 2011

wcsss → actual - predicted (error).

June 2011

Wk	Mo	Tu	We	Th	Fr	Sa	Su
22							
23	6	7	1	2	3	4	5
24	13	14	8	9	10	11	12
25	20	21	15	16	17	18	19
26	27	28	22	23	24	25	26
			29	30			

Week 23 ▪ 161-204

* Ploting silhouette score →

```
k = list(range(2,11))
ys = []
for i in k:
    kmeans = KMeans(n_clusters=i, random_state=1)
    kmeans = fit(df)
    labels = kmeans.labels_
    ys.append(silhouettescore(df, labels))
```

```
plt.plot(k, ys, color='m')
plt.xlabel('k')
plt.ylabel('silhouette score')
```

PCA V/S LDA

PCA → Principal component Analysis

LDA → Linear Discriminant Analysis

- Feature selection technique / Dimensionality Reduction.

Target Based Problem (for supervised learning) → use LDA to select significant features.
(LDA is specifically designed for supervised learning)

Target less. (for unsupervised) → use PCA to get important feature.
(PCA can be used for supervised and unsupervised.)
But designed for unsupervised

- LDA, PCA makes model complex.
  We dont use it much.
  (Try to avoid LDA, PCA in real time).
  → If Performance increases then use it

Saturday 11
June 2011
Week 23 ▪ 162-203

LDA.

Step 1. computing d-dimensional mean vectors.
Step 2. computing scatter matrices.
→ within class scatter matrix
→ between class scatter matrix
Step 3. Find eigen values & eigen vectors. by linear algebra packages.
Step 4. calculate ratio of eigen values.
Step 5. eliminate non-significant features.
& evaluate with significant ones.

```
from sklearn.discrimant_analysis import
                LinearDiscriminantAnalysis as LDA
lda = LDA(n_components=2)
        └──→ take top2 eigen values.
             (with highest %.)
lda.fit_transform(x,y)
```

After this give this transformed value to any algorithm; ~~LDA~~ logistic, linear etc. (it will give my 2 feature to algo → you relive selected 2) ∴ training time will be less.

Sunday 12
163-202

PCA.

all the steps are same as LDA, only difference is in step2.
→ It will not compute 2 diffrent scatter matrix as we don't have target defined in unsuperv.

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 2).
X_pca = pca.fit_transform(X)
```

13 Monday
June 2011

Week 24 ▪ 164-201

Day-31 REVISION.

- **Unsupervised learning -**
  - It is a class of ml technique to find pattern in data.,
  - The data given is not labeled. (no Target(y))
  - Patterns in data are used to identify / group similar observation.

Clustering

- It discovers hidden structures of data.
- A way to decompose set into subset, with each subset representing group with similar characteristics.
- These groups → are called clusters..

Clustering

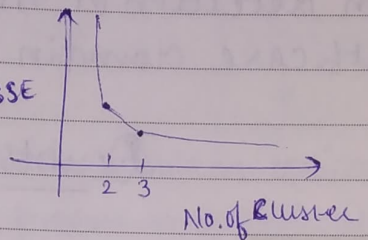
- Top-down approach → connectivity based clustering
  - Ex. Hierarchical
- centroid based clustering.
  - Ex. K-means *

K-Means

1. Select random points as centroids.
2. Calculate distance.
   → Distance being calculated by 3 methods
   - $|x_2 - x_1| + |y_2 - y_1|$ → manhattan
   - $\sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$ → Euclidean (Preffered) *
   - $\max(|x_2 - x_1|, |y_1 - y_2|)$ → chebyshev

③ Update centroids by taking the average of its data points
④ repeat step ② & step ③ till distance of each data point is nearer to the own cluster mean than the other cluster. } termination condition.

- Elbow Method.
  - ↳ To find optimum k-value.



- Performance Estimation :- (Silhouette-coefficient)
  - It is a measure of how similar a data point is to its own cluster compared to that of other clusters.
  - lies b/w −1 to +1

How it is calculated

$$s(i) = \begin{cases} 1 - \frac{a(i)}{b(i)} & \text{if } a(i) < b(i) \\ 0 & \text{if } a(i) = b(i) \\ \frac{b(i)}{a(i)} - 1 & \text{if } a(i) > b(i) \end{cases}$$

Here,
$a(i)$ → dis of individual pt.
$b(i)$ → mean value of centroids

Pros:
→ With large var., k means is faster than other clustering algo.
→ Produces tighter cluster than Hierarchical.
→ Less impacted by outliers

Cons:
→ Predicting no. of clusters is a tedious task
→ If data has original clusters, the algo. doesn't work well

- Img. & video tracking → kmeans is best.

Industry Applications →

1. Customer segmentation.
2. Anomaly Detection
3. Creating newsfeeds. (cluster articles, based on similarity)
4. Pattern detection in medical imaging for diagnostics.
   → Healthcare domain.

Doubts-clearing session.

In Naive Bayes,

- Bernoulli → when DV is Binary categorical
- multinomial → more than 2 categories in DV.
- Gaussion → when numerical values converted into range.

Precision → zscore/ztest
Fscore → Ftest
Recall → Ttest

} as we did in statistical learning, these values are same.

Ensemble Technique.

Day-32.

Agenda.

- Intro. to Ensemble Techniques.
- Bagging
- Boosting.
- Stacking
- case study on classification problem optimization with ensemble
- case study on Regression problem optimization with ensemble
- How to take forward this production.

Ensemble Techniques / Methods:

Aggregating
divide & aggregate all the soln of task & then come with a soln.)

- Ensemble is a meta-algorithms that combine several ML techniques into one predictive model in order to decrease (1) variance (bagging), (2) bias (Boosting), (3) improve prediction (Stacking).

→ To improve performance of model, Ensemble is used.

(2) bias (Boosting) → assumption (objective is that, assumption will be close to outcome)

* Bias should be minimized

(3) improve prediction (Stacking) → actual-pred. (objective is to minimize variance).

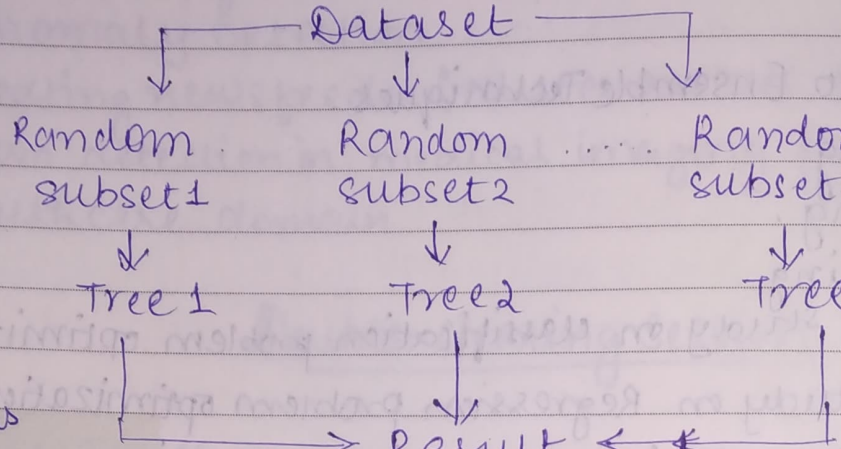
17 Friday

June 2011

Week 24 ▪ 168-197

Bagging Ensemble.

→ It converts the dataset into randomly n subsets, so that every subset include every type of data.



Ques. What algo. is applied inside ensemble?

The Base model is running inside it, ie if you train your model with logistic regression & then applied ensemble so, the logistic regr. is applied to every subset.

* Exception →

Random Forest is not applicable in Bagging, as it itself is a Ensemble Technique.

Bagging V/s Boosting.

Bagging

If model is split into 10 subsample, for all 10 subsamples, Model is Built/Trained at a time. (Parallel Model construction)

∴, Training time is very less.

Bootstrap aggregation ↓ which invokes automatically. (autom. it divides data)

Boosting.

↳ It is not parallel execution, it is sequential execution.

Explanation.

Boosting.

ex. If the dataset is split into 10 records - then the 1st model is trained, once the model is Built or trained, the result of 1st model goes to model2.
(If Base algo. is logistic, then what hyperparameters are used by model1., what was the error rate, why it is getting that error..., are the questions model2 takes.)
& using the model1 output as a inference, it trains the model2...., this process continues till Best model9 is trained10,
it goes sequentially.,
the final outcome is far better from model1.
(Best model trained).

∴, training time is very high as till the time 1st model is training model2 has to wait.
∴ nmodel has to wait till completion of n-1 model.

Stacking/Voting.

↳ calling multiple algorithms for a same dataset & then find which algo. is giving Best accuracy on dataset.

Sunday 19
170-195

ex. If dataset is split into 10 subsamples,
in first time, logistic is applied to all 10 samples,
out of which 7 are saying/giving accuracy 1.
∴, for logistic model it will take the no. which is getting highest no. of votes (Here 1)
then D.T is performed... similar process goes on for all 10 samples.

(It is concerned about majority of samples.)

20

Monday

June 2011

Except RandomForest, ExtraTree all algo. can use these Ensemble techniques.

* use voting for final predn.

use Avg of all preds

Classification	Regression
• BaggingClassifier()	• BaggingRegressor()
• RandomForestClassifier()	• RandomForestRegressor()
• ExtraTreesClassifier()	• ExtraTreesRegressor()

• **Bagging** → splits data into n subsamples
→ create model (& the samples are choosen with replacement.)
the original dataset remains same.

without replacement
↓
original dataset is not same
ex If model1 choose 100 records from dataset, Now dataset is left with 900 records, model2 will choose samples from these 900 records.

• **Random Forest**
→ It follows bagging technique but it selects a set of features which are used to decide best split at each node of DT.
→ Base estimator is D.T.
→ You can use pruning. (max depth, ... etc) can be used while classifying.

• **Extra Trees** (Extremely Randomized Trees).
→ You can't use pruning, this is unpruned algorithm.
→

XGBoost → (Extreme Boosting / Gradient)
ie, it will go till ~~maximum~~ Global minima.

Tuesday 21 June 2011

Week 25 ▪ 172-193

Boosting Algorithms:

Classification	Regression
• AdaBoostClassifier()	• AdaBoostRegressor()
• GradientBoostingClassifier()	• GradientBoostingRegressor()

Adapting Boosting.
(Based on week learner current model is updating the weights ie it is Adapted.)

GradientBoost
(It is a pt. where we encounter cost fn)
→ In this Gradients are trying to be minimized based on previous model.

Practical Implementation.
→ Dataset ('Pima Diabetes').

```
df = pd.read_csv('diabetes.csv')
df.head()
```

- split into x & y.

```
x = df.drop('outcome', axis=1)
y = df.outcome
```

- split into train & test data.
- Build logistic Model & find accuracy → ~78%.
- Build KNN model & find accuracy. → ~74%.
  (Ex. K=3)

* Bagging.

```
from sklearn.ensemble import BaggingClassifier
bg = BaggingClassifier(base_estimator = log_reg,
                       n_estimator = 12,
                       verbose = 1)

bg.fit(xtrain, ytrain)
bgpred = bg.predict(xtest)
```

(log_reg → logistic reg. function defined earlier)

Traing score.

```
bg.score(xtrain, ytrain)
```

Testing score.

```
bg.score(xtest, ytest)
```

Accuracy.

```
accuracy_score(ytest, bgpred)        ~ 78%.
```

(there was no change in logistic regression acc., so don't include baging classifier for this dataset).

Day-33.

* Randomforest.

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators = 10)
rf.fit(xtrain, ytrain)
rf_pred = rf.predict(xtest)

accuracy_score(ytest, rf_pred)        → 71%
```

finetune the hyperparameters.

```
rfl = RandomForestClassifier(n_estimators=8, max_depth=5,
                             max_features=6, random_state=2,
                             verbose=2)
rfl.fit(xtrain, ytrain)
rfl_pred = rfl.predict(xtest)
```

In this we have 8 features, so select Top6 only.

~77% `accuracy_score(ytest, rfl_pred)`

from previous model the acc. increases from 71 to 77 %.

* ExtraTree Ensemble.

```
from sklearn.ensemble import ExtraTreesClassifier
et = ExtraTreesClassifier(n_estimators=8)
et.fit(xtrain, ytrain)
et_pred = et.predict(xtest)
accuracy_score(ytest, et_pred)  → 73%.
```

- when you increase to 10, accuracy is 75%.
- with → 12 ∴ acc = 77%.

* Overfitting & Underfitting → Test ↑ Train ↓.

Train ↑
Test ↓

`rf.score(xtrain, ytrain)` → Training accuracy.
`rf.score(xtest, ytest)` → Test accuracy

① High variance
② more outlier

* since, the data is imbalanced, so there is high chances of overfitted data model.

0 → 500
1 → 268

Not equal 0's & 1's ∴, imbalanced dataset.

upsampling/downsampling.

Q. How to overcome this problem?
You may use balancing technique to balance dataset,
either 268 pushes to 500 or 500 pulls down to 268.
This Balancing might reduce/overcome overfitting problem.

Boosting.

ADA BOOST

```
from sklearn.ensemble import AdaBoostclassifier
ada = AdaBoostclassifier(best_estimator=log, n_estimators=10)
ada.fit(xtrain, ytrain)
ada_pred = ada.predict(xtest)
accuracy_score(ytest, ada_pred)
```

log → Logistic regres.

→ 67%.

Gradient BOOST.

```
from sklearn.ensemble import GradientBoostingClassifier
gb = GradientBoostingclassifier(n_estimators=10)
gb.fit(xtrain, ytrain)
gbpred = gb.predict(xtest)
accuracy_score(ytest, gbpred)
```

→ 77%.