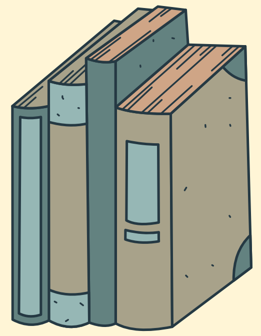
# MACHINE LEARNING WEEK #3

KAMPUS MERDEKA X MYEDUSOLVE

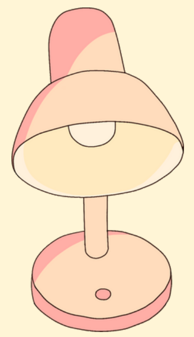
Team 3 - Data Science A

# MACHINE LEARNING



SUPERVISED LEARNING

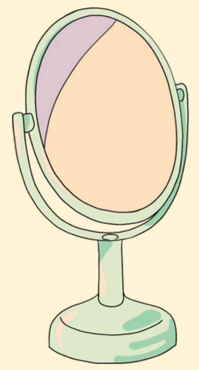
HYPERPARAMETER



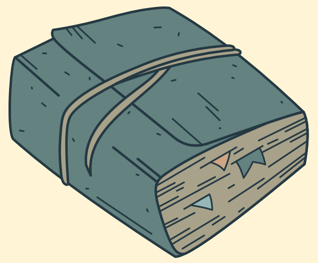
MODEL OPTIMIZATION

PIPELINE

IMBALANCED DATA



ENSEMBLE MODEL



UNSUPERVISED LEARNING

## SUPERVISED LEARNING

Supervised learning is a machine learning approach that uses labeled data or datasets that are already known to the engineer.

## CLASSIFICATION

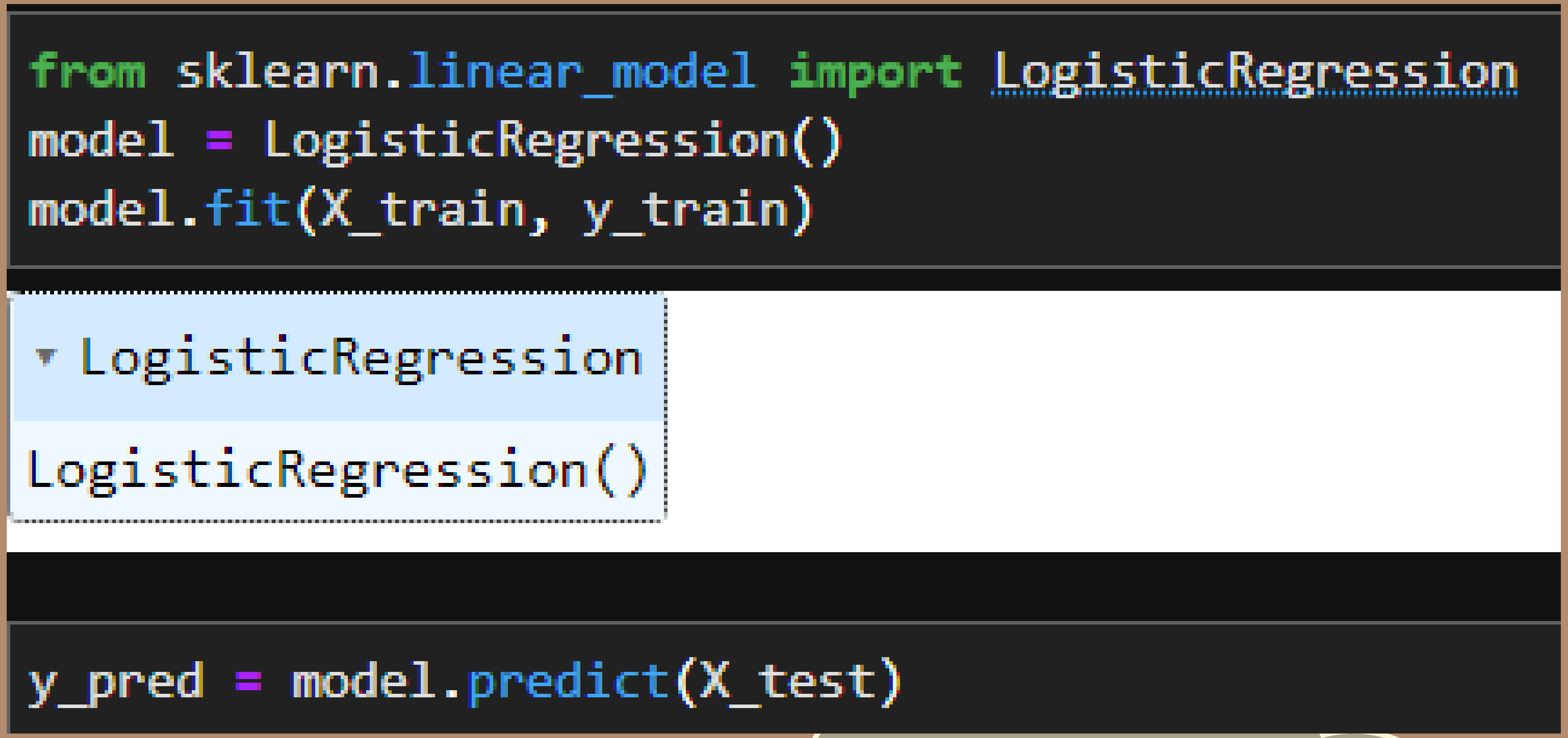
Classification is the process of **grouping** data into several categories to make it easier to process and analyze.

The **evaluation model** used is Precision-Recall, ROC-AUC, Accuracy, Log-Loss

# LOGISTIC REGRESSION

Logistic Regression is a classification algorithm to find the relationship between discrete features and the probability of certain discrete output results with basic linear regression calculations.

## How to use

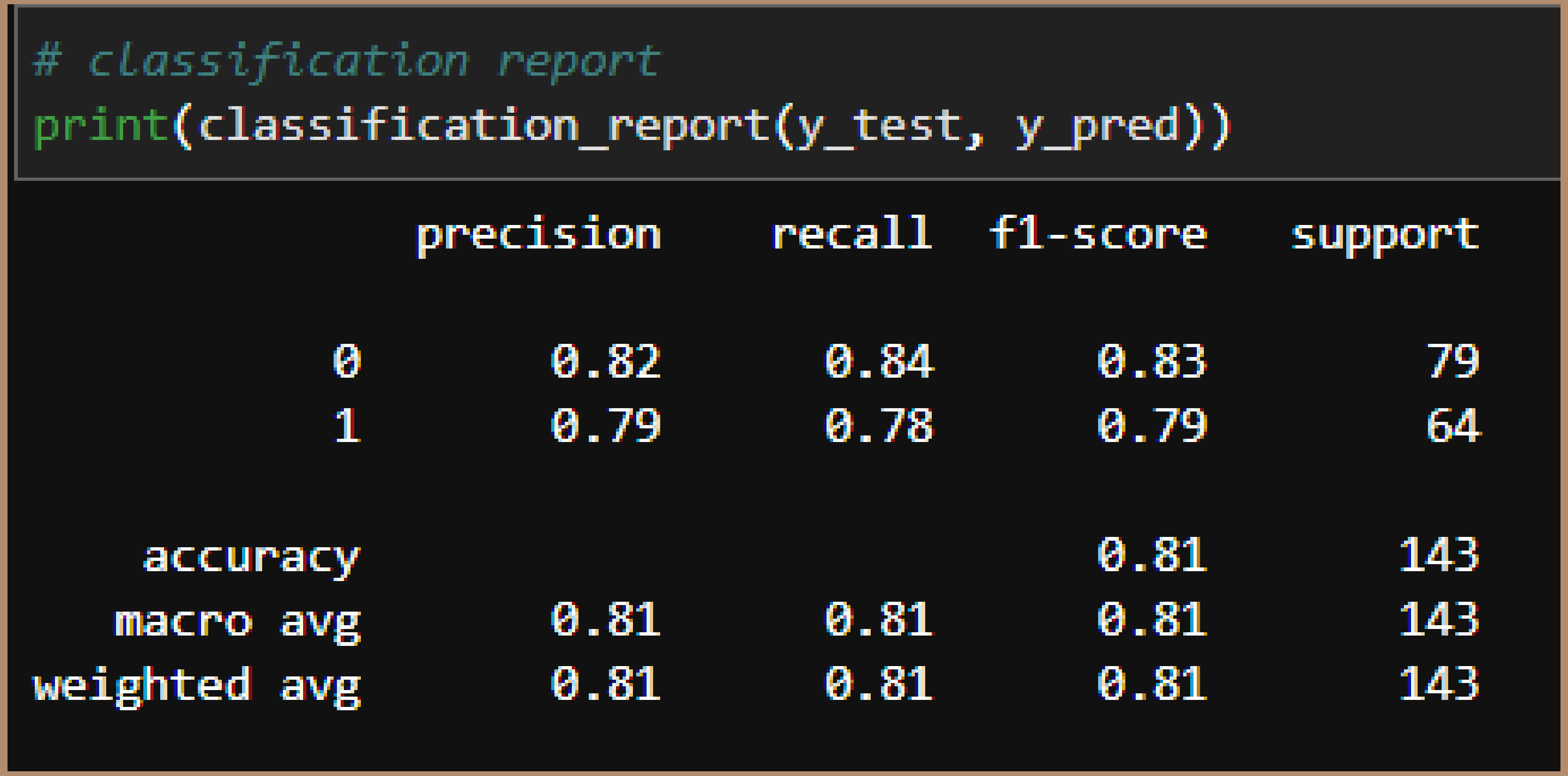


```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
y_pred = model.predict(X_test)
```

## Model Evaluation



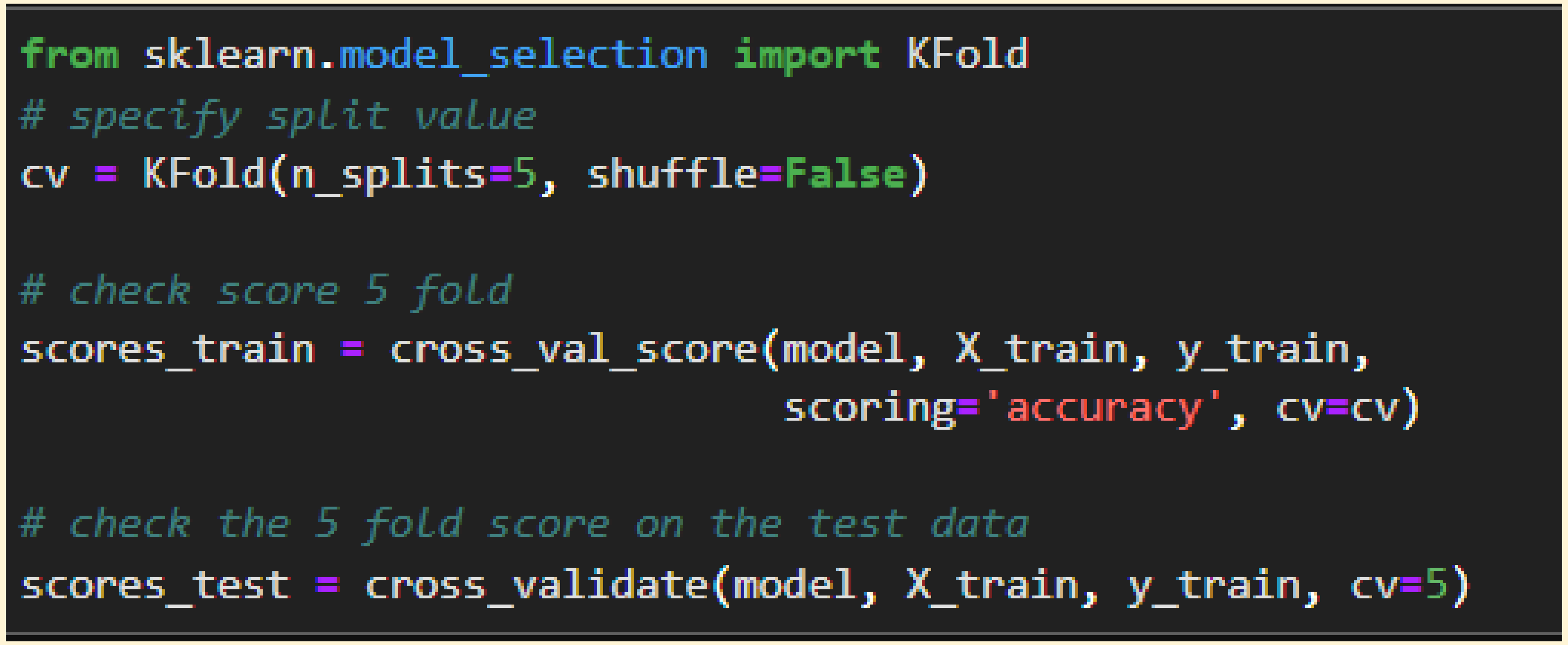
```
# classification report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.84      0.83        79
           1       0.79      0.78      0.79        64

    accuracy                           0.81       143
   macro avg       0.81      0.81      0.81       143
weighted avg       0.81      0.81      0.81       143
```

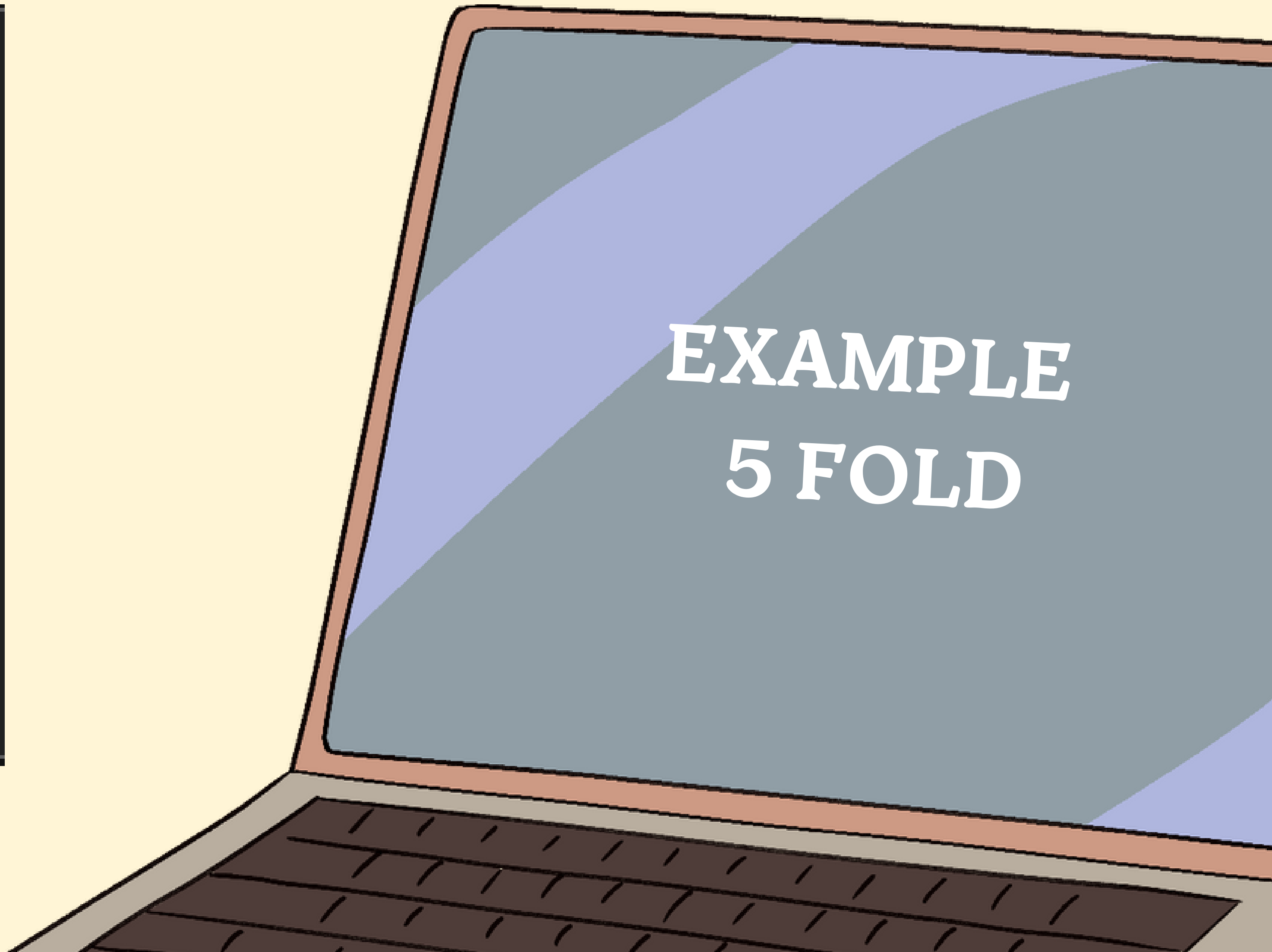
# MODEL OPTIMIZATION

**Cross Validation**

Cross Validation is a cross-validation technique that divides a dataset into two parts which are called training data and test data. This is done by dividing the data into various partitions. That is why cross-validation is also often called K-Fold Cross Validation because where the experiment is carried out a total of k values.

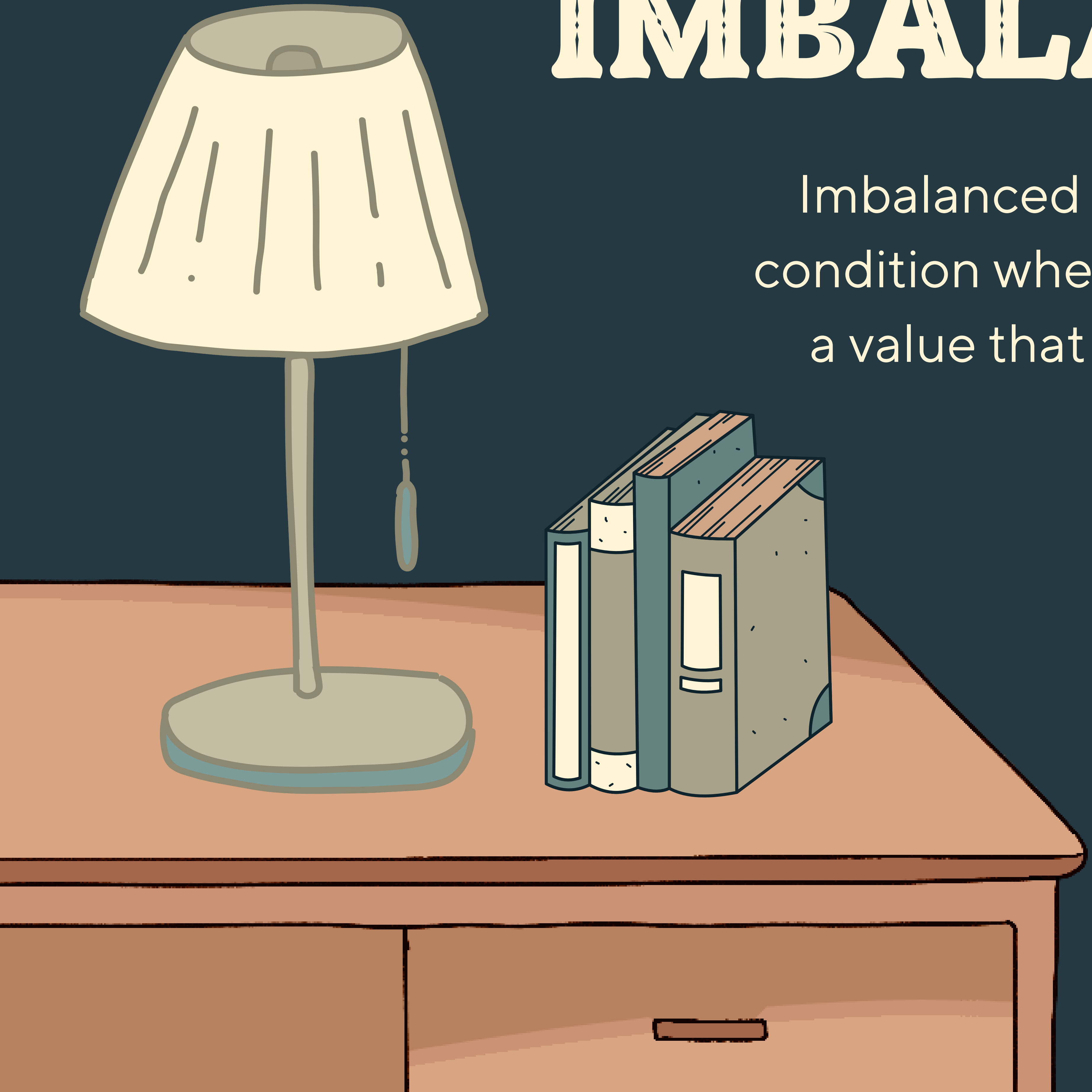


```python
from sklearn.model_selection import KFold
# specify split value
cv = KFold(n_splits=5, shuffle=False)

# check score 5 fold
scores_train = cross_val_score(model, X_train, y_train,
                               scoring='accuracy', cv=cv)

# check the 5 fold score on the test data
scores_test = cross_validate(model, X_train, y_train, cv=5)
```



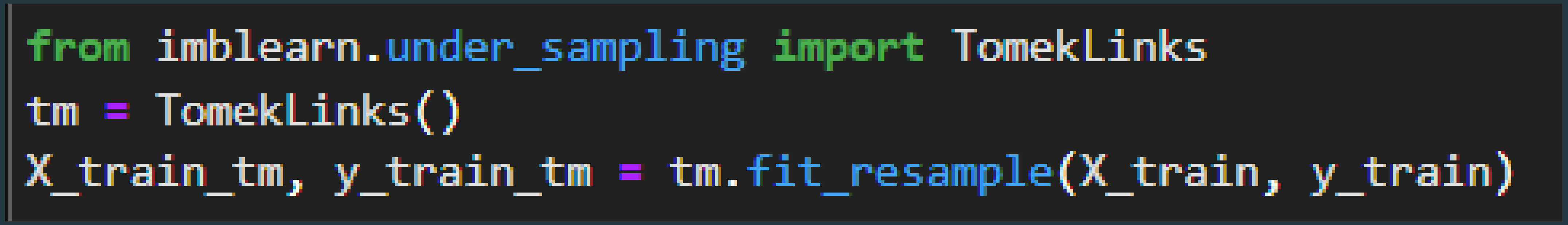
# IMBALANCED DATA

Imbalanced data is a normal data state, with a condition where one of the labels on the data has a value that is very much different in number from other classes.



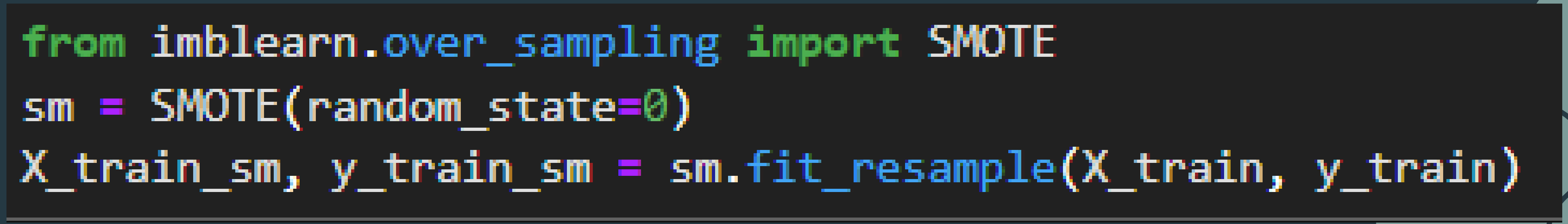
# HANDLING IMBALANCED DATA

**Undersampling:** keep minority, decrease majority data



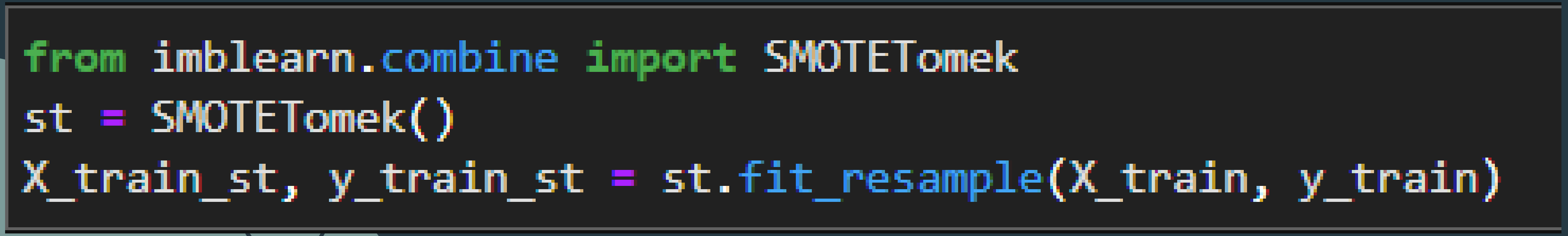
```python
from imblearn.under_sampling import TomekLinks
tm = TomekLinks()
X_train_tm, y_train_tm = tm.fit_resample(X_train, y_train)
```

**Oversampling:** keep minority, decrease majority data



```python
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=0)
X_train_sm, y_train_sm = sm.fit_resample(X_train, y_train)
```

**Combine:** SMOTE and TomekLink combined



```python
from imblearn.combine import SMOTETomek
st = SMOTETomek()
X_train_st, y_train_st = st.fit_resample(X_train, y_train)
```

## PARAMETER

Parameters are the default values of the model's internal configuration whose values we cannot change because they are directly related to the algorithm of the model itself.

## HYPERPARAMETER

Hyperparameters are external parameters of the model whose values we can change and set manually before training the model.

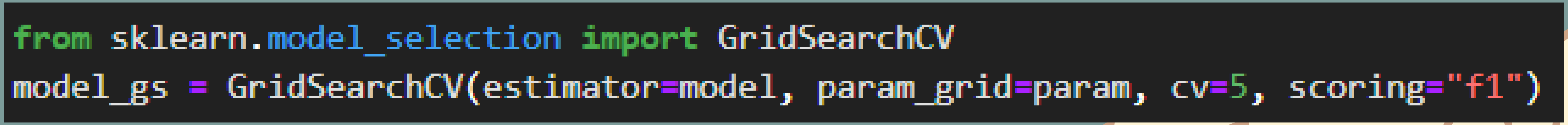
## HYPERPARAMETER TUNING

**Manual Tuning**

This process relies on previous experience in manually searching for the best hyperparameter optimization.

**Grid Search**

Grid Search will search for all combinations of all the parameters that we set and then will choose the best combination based on the value of the highest mean CV score.

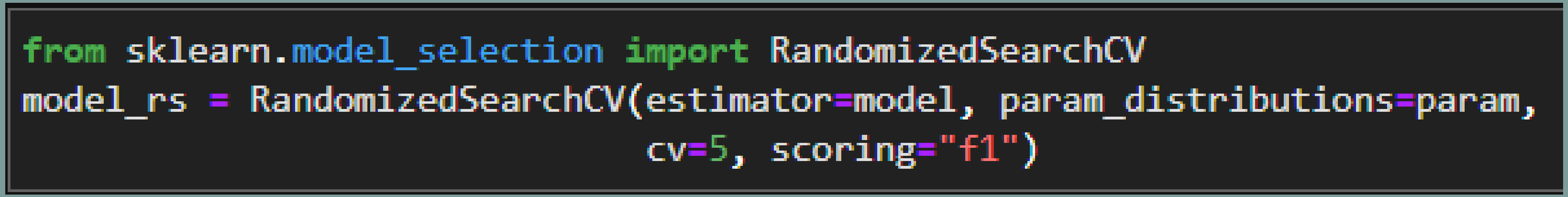


```
from sklearn.model_selection import GridSearchCV
model_gs = GridSearchCV(estimator=model, param_grid=param, cv=5, scoring="f1")
```

# HYPERPARAMETER TUNING

## Random Search

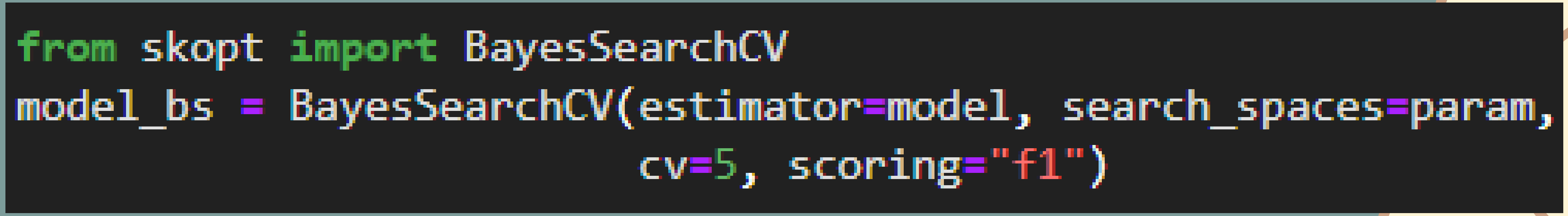
Random Search is similar to Grid Search, but Random Search will perform randomly, so that in the end we get a "representative" combination of results.



```
from sklearn.model_selection import RandomizedSearchCV
model_rs = RandomizedSearchCV(estimator=model, param_distributions=param,
                              cv=5, scoring="f1")
```

## Bayesian Optimization

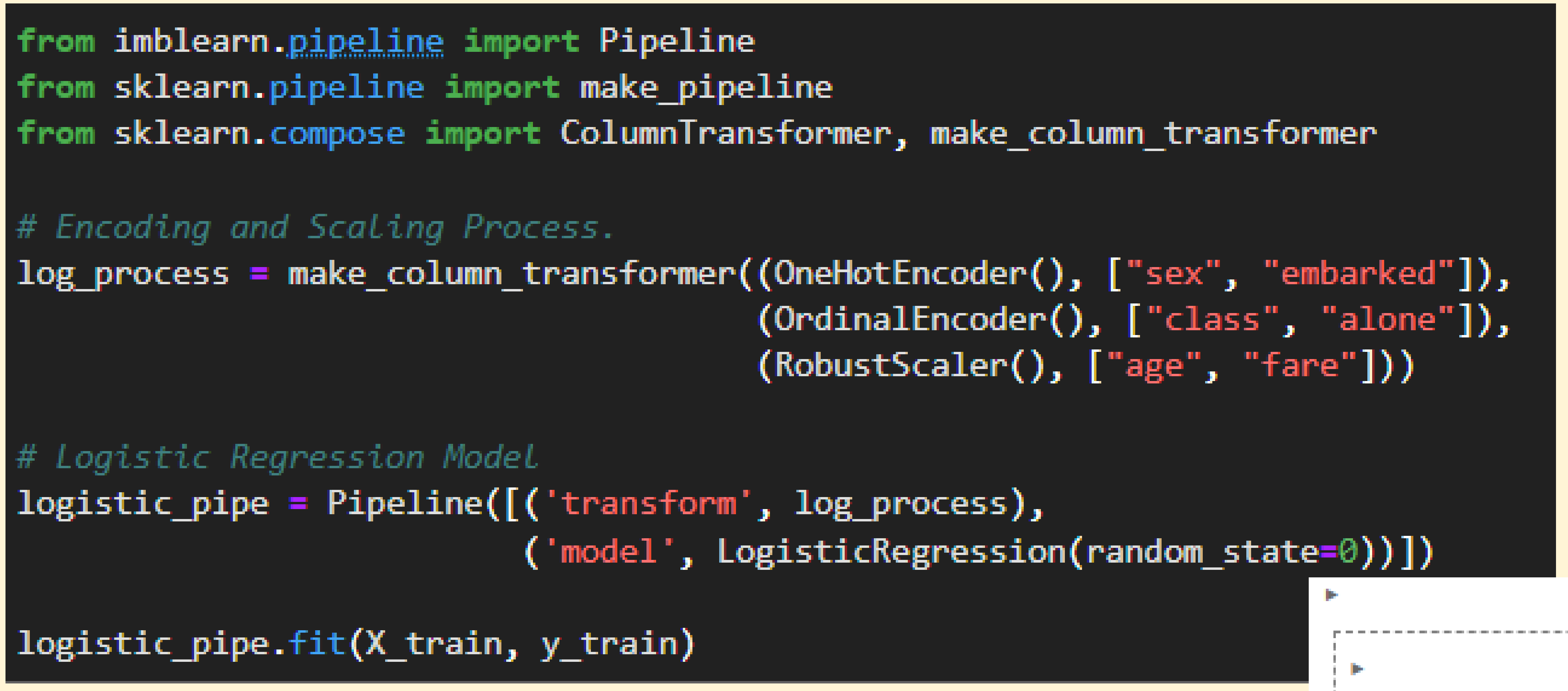
Bayesian Optimization uses the Bayes Theorem principle, where optimization is carried out based on probability.



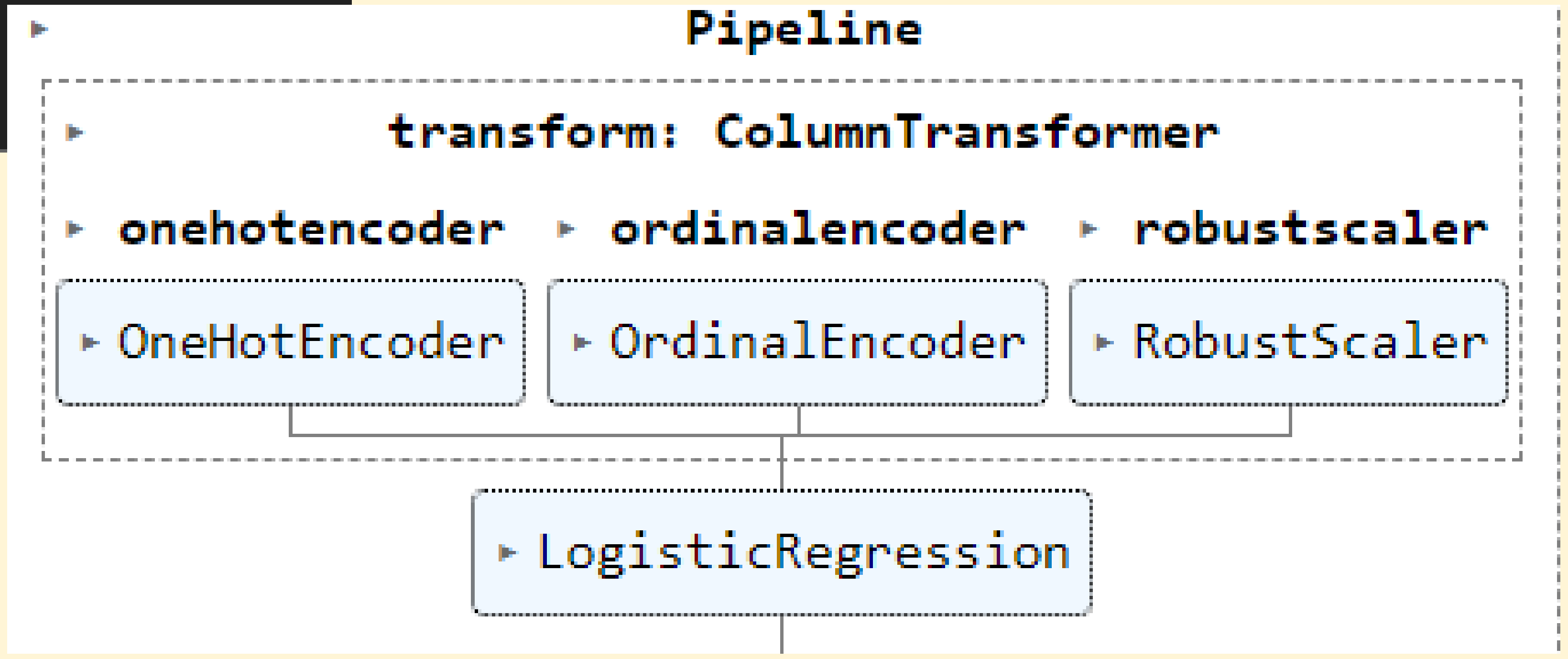
```
from skopt import BayesSearchCV
model_bs = BayesSearchCV(estimator=model, search_spaces=param,
                         cv=5, scoring="f1")
```

## PIPELINE

pipeline is a series of code commands that can be executed independently to run all tasks and processes involved in modeling, from data preprocessing to model evaluation.



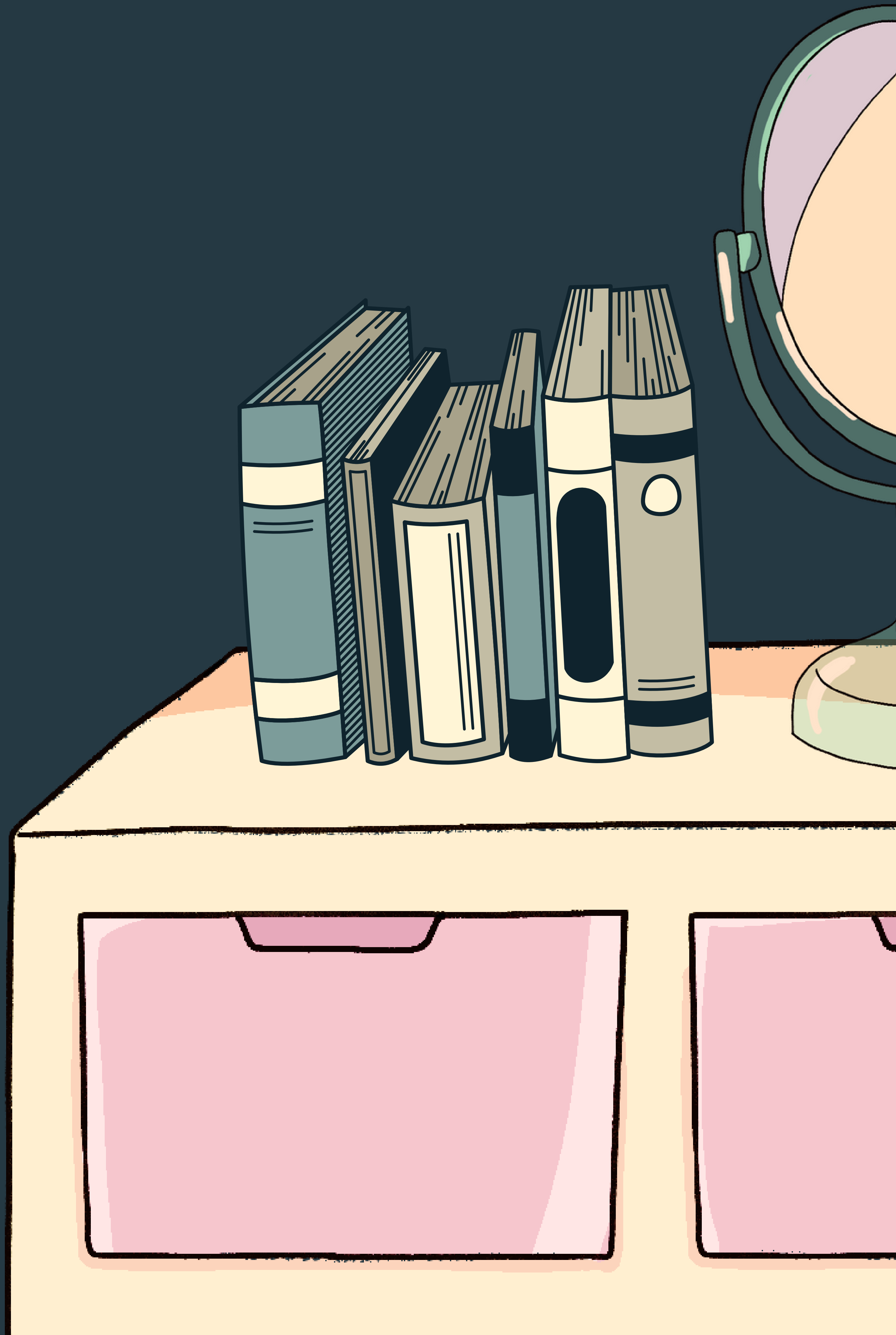
```python
from imblearn.pipeline import Pipeline
from sklearn.pipeline import make_pipeline
from sklearn.compose import ColumnTransformer, make_column_transformer

# Encoding and Scaling Process.
log_process = make_column_transformer((OneHotEncoder(), ["sex", "embarked"]),
                                      (OrdinalEncoder(), ["class", "alone"]),
                                      (RobustScaler(), ["age", "fare"]))

# Logistic Regression Model
logistic_pipe = Pipeline([('transform', log_process),
                          ('model', LogisticRegression(random_state=0))])

logistic_pipe.fit(X_train, y_train)
```



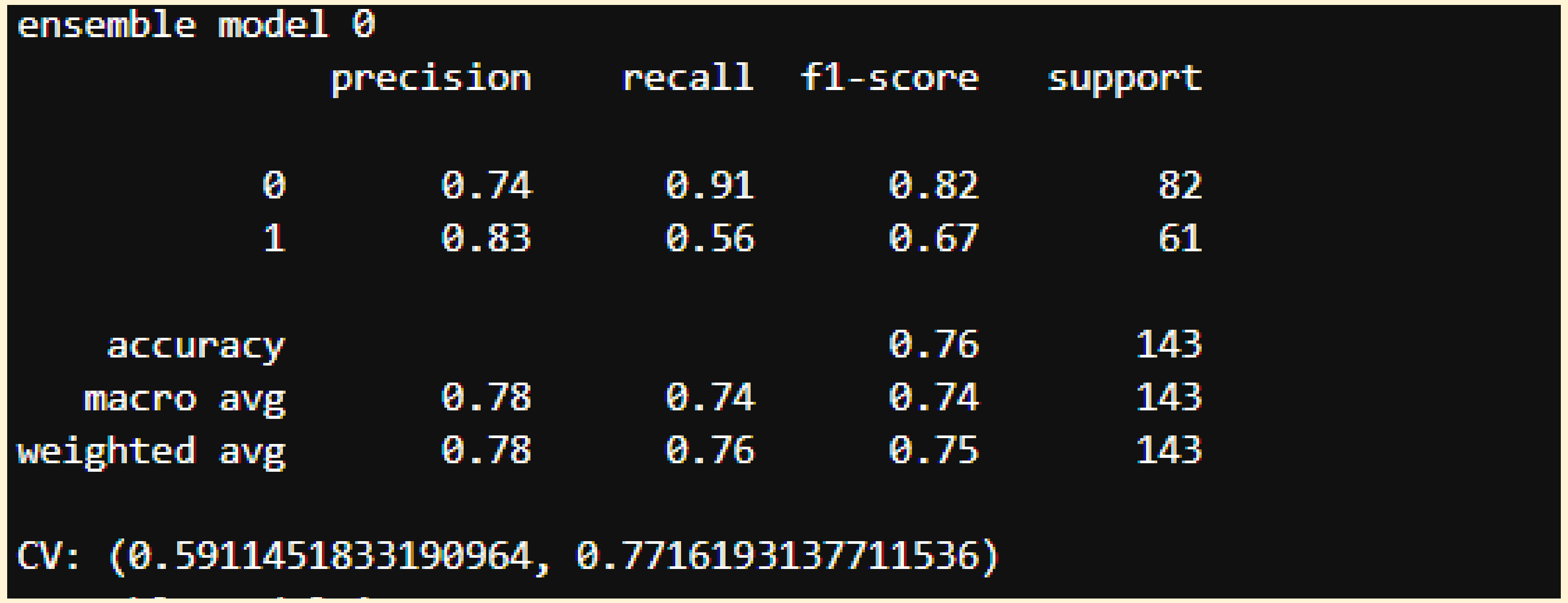
## ENSEMBLE MODEL

The way algorithm learns the data is by using a combination of several algorithms/models to get output with higher accuracy when compared to using only one model/algorithm.

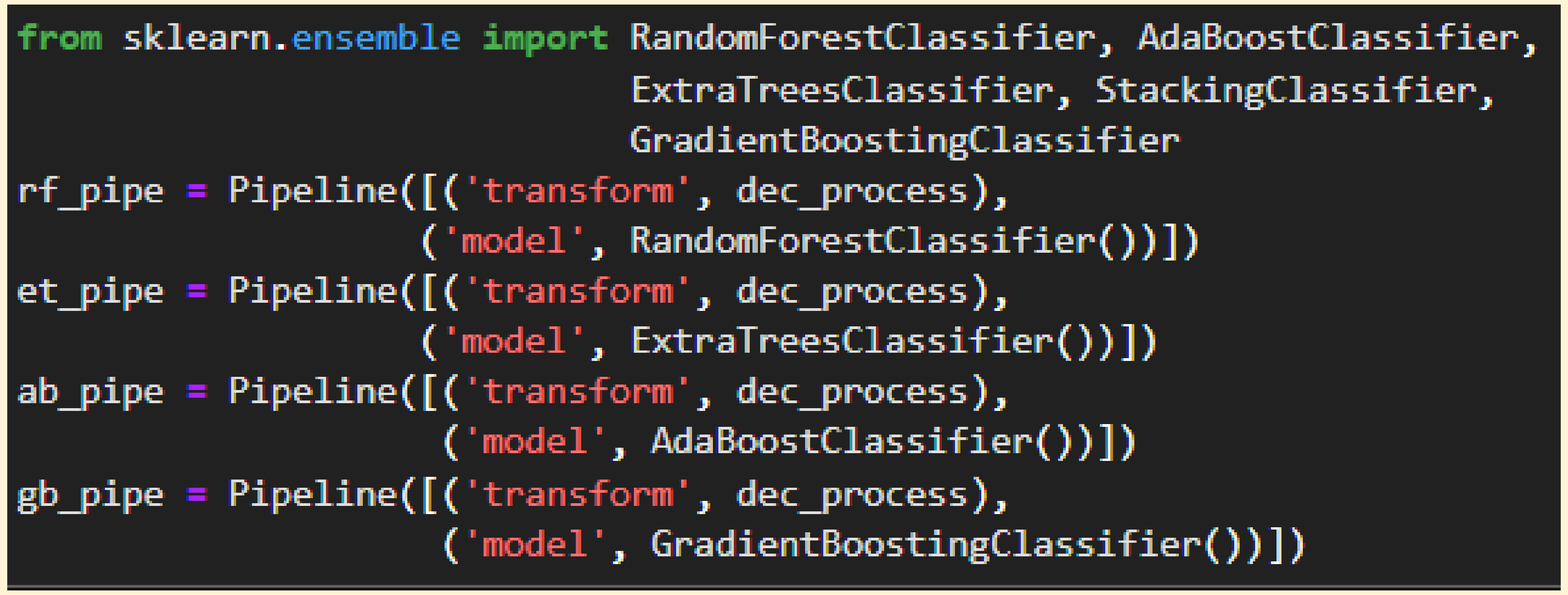
There are many ways to use an ensemble, but in general, the algorithm process is averaging, voting, and weighting which is generally divided into Bagging, Boosting, and Stacking.



## CLASSIFICATION REPORT (RANDOM FOREST)



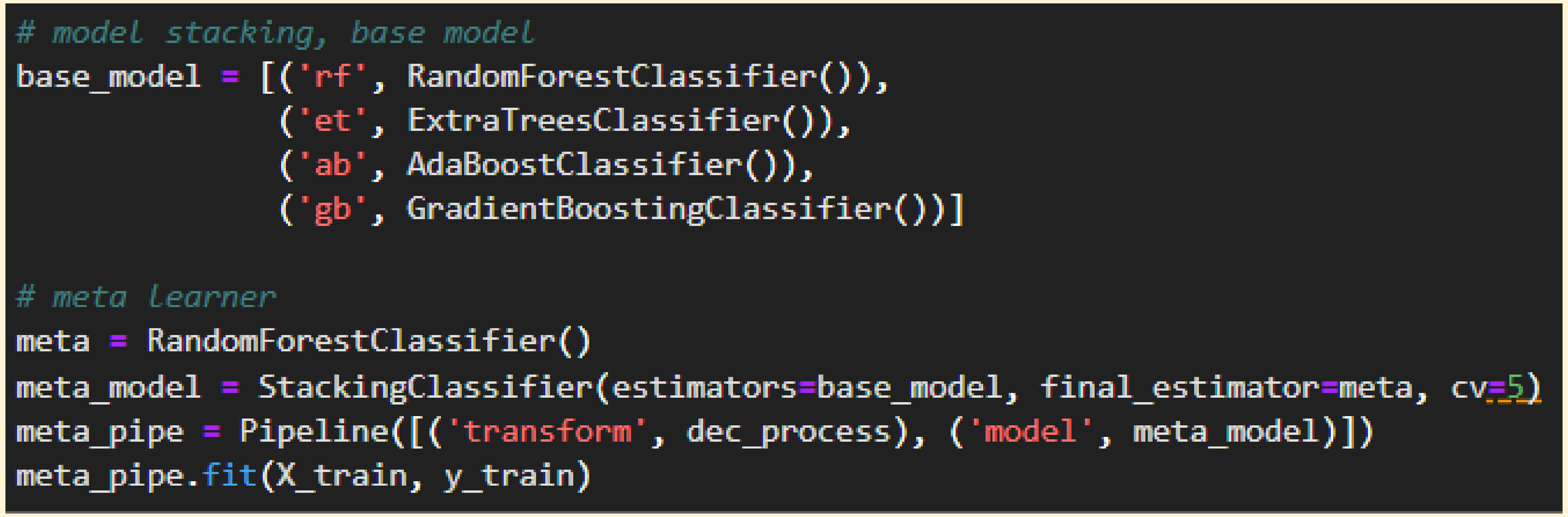
```
ensemble model 0
              precision    recall  f1-score   support

           0       0.74      0.91      0.82        82
           1       0.83      0.56      0.67        61

    accuracy                           0.76       143
   macro avg       0.78      0.74      0.74       143
weighted avg       0.78      0.76      0.75       143

CV: (0.5911451833190964, 0.7716193137711536)
```



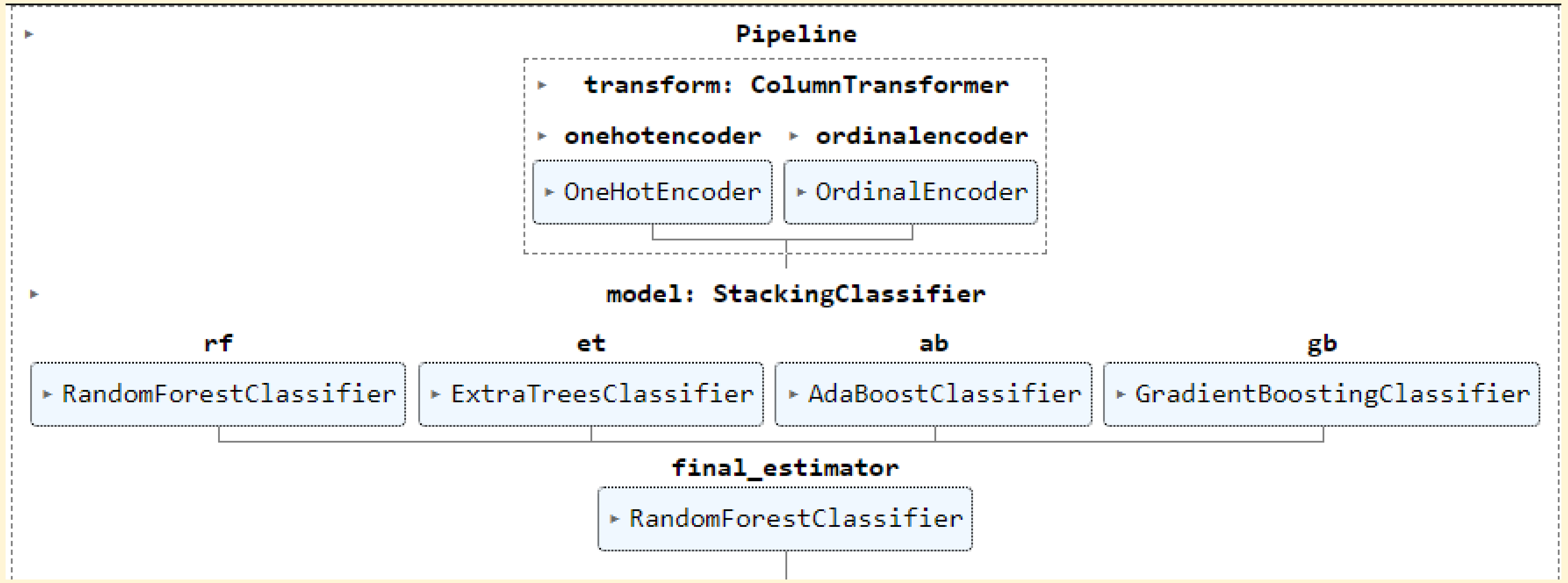
```python
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier,
                             ExtraTreesClassifier, StackingClassifier,
                             GradientBoostingClassifier
rf_pipe = Pipeline([('transform', dec_process),
                   ('model', RandomForestClassifier())])
et_pipe = Pipeline([('transform', dec_process),
                   ('model', ExtraTreesClassifier())])
ab_pipe = Pipeline([('transform', dec_process),
                    ('model', AdaBoostClassifier())])
gb_pipe = Pipeline([('transform', dec_process),
                    ('model', GradientBoostingClassifier())])
```

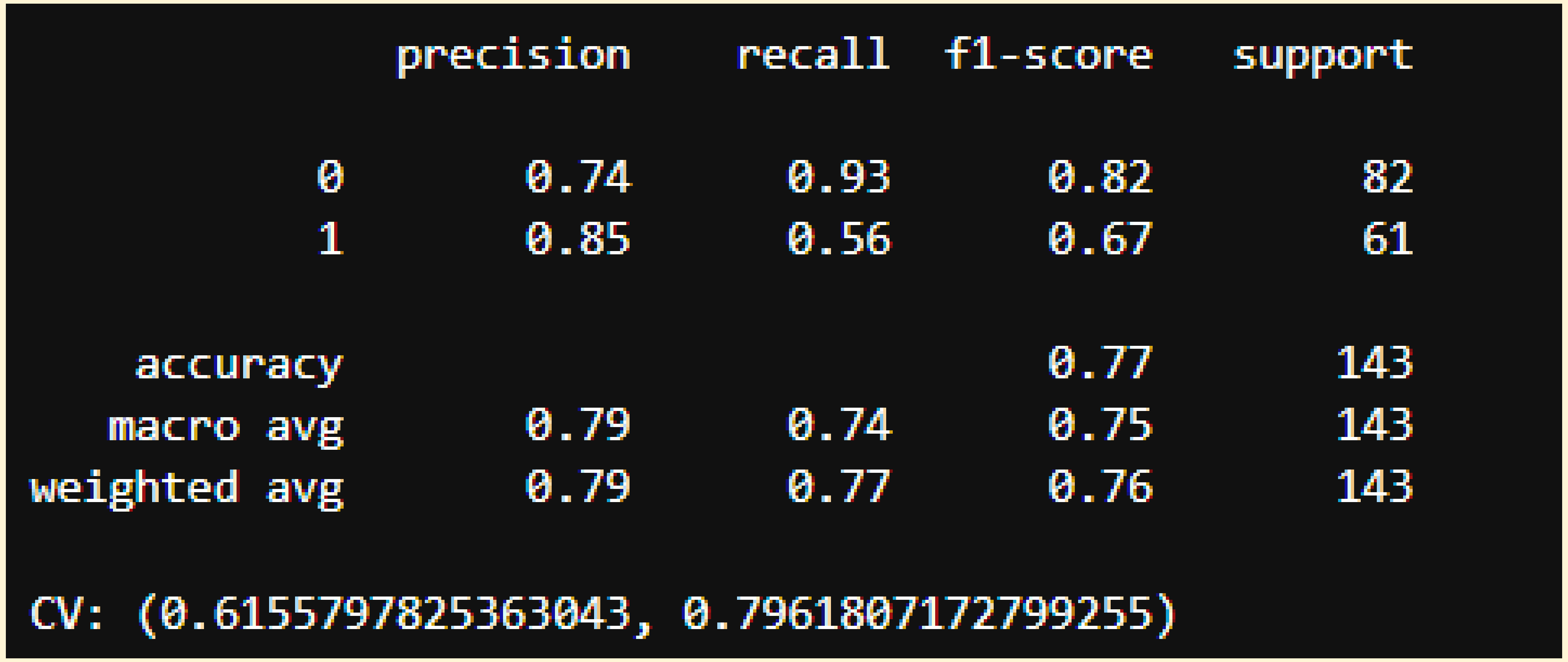
## USING PIPELINE

## META LEARNER



```python
# model stacking, base model
base_model = [('rf', RandomForestClassifier()),
              ('et', ExtraTreesClassifier()),
              ('ab', AdaBoostClassifier()),
              ('gb', GradientBoostingClassifier())]

# meta learner
meta = RandomForestClassifier()
meta_model = StackingClassifier(estimators=base_model, final_estimator=meta, cv=5)
meta_pipe = Pipeline([('transform', dec_process), ('model', meta_model)])
meta_pipe.fit(X_train, y_train)
```





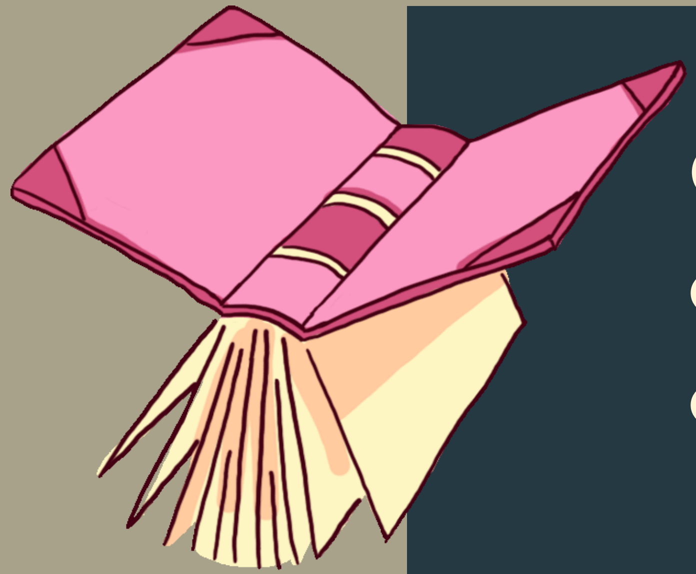
```
              precision    recall  f1-score   support

           0       0.74      0.93      0.82        82
           1       0.85      0.56      0.67        61

    accuracy                           0.77       143
   macro avg       0.79      0.74      0.75       143
weighted avg       0.79      0.77      0.76       143

CV: (0.6155797825363043, 0.7961807172799255)
```

## CLASSIFICATION REPORT (META LEARNER)

## UNSUPERVISED LEARNING

Unsupervised Learning allows users to discover intrinsic patterns from data and perform more complex processing tasks than supervised learning.

# CLUSTERING

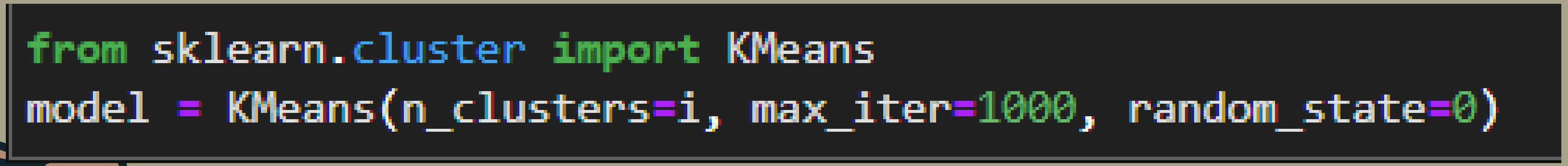


Clustering is the process of partitioning a set of data objects into sets called clusters. Objects in the cluster will have similarities with each other and will be different from other clusters.

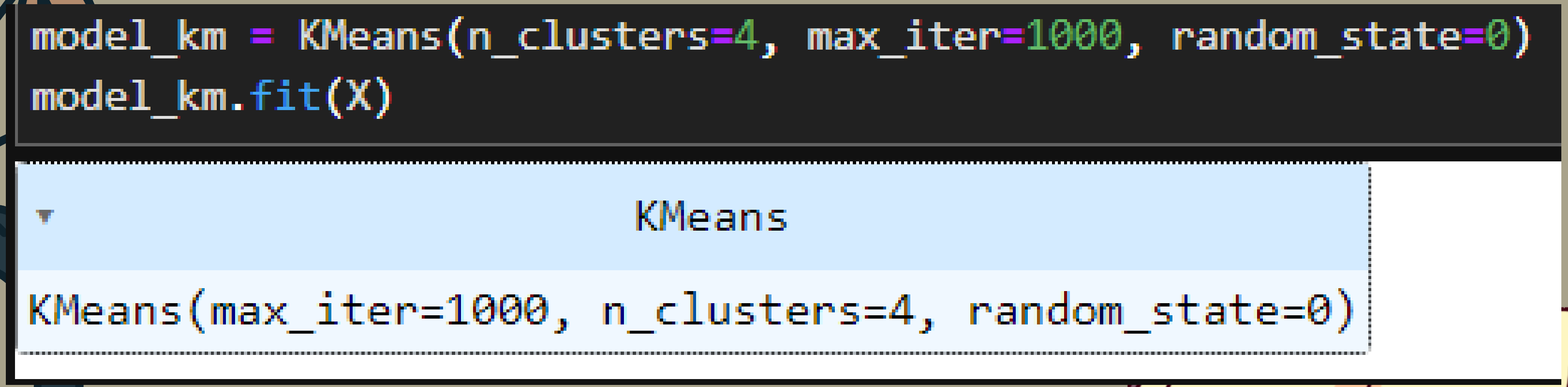
An example of a clustering model is
1. Partitional - K-Means
2. Density Based - DBSCAN
3. Hierarchical - Agglomerative Clustering

# K-MEANS

K-Means Clustering is a data analysis method that performs an unsupervised learning modeling process and uses a method that groups data into various partitions.

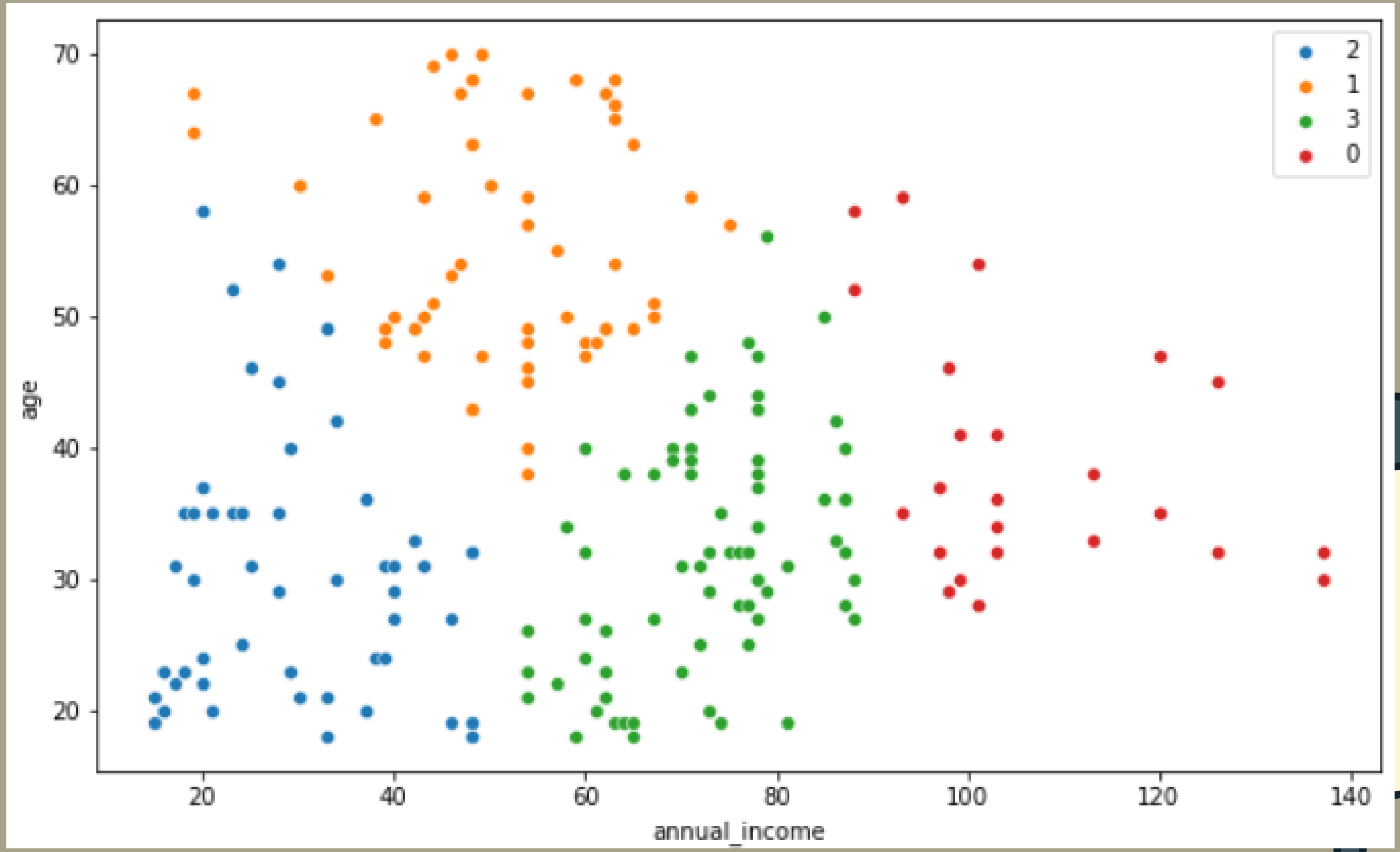


```
from sklearn.cluster import KMeans
model = KMeans(n_clusters=i, max_iter=1000, random_state=0)
```



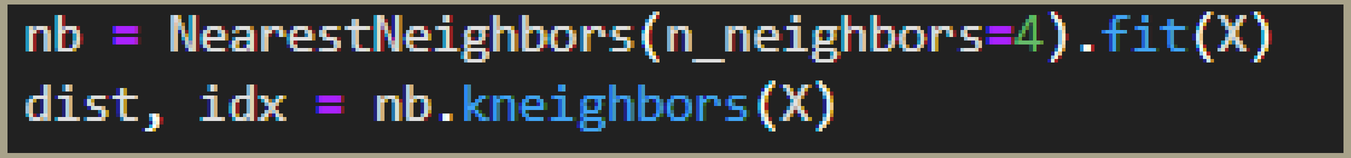
```
model_km = KMeans(n_clusters=4, max_iter=1000, random_state=0)
model_km.fit(X)
```

```
▾ KMeans
KMeans(max_iter=1000, n_clusters=4, random_state=0)
```

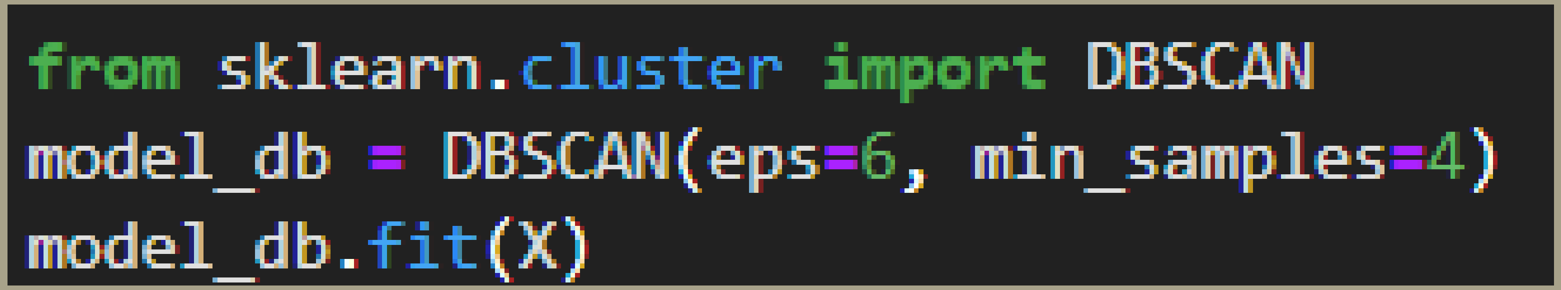


# DBSCAN

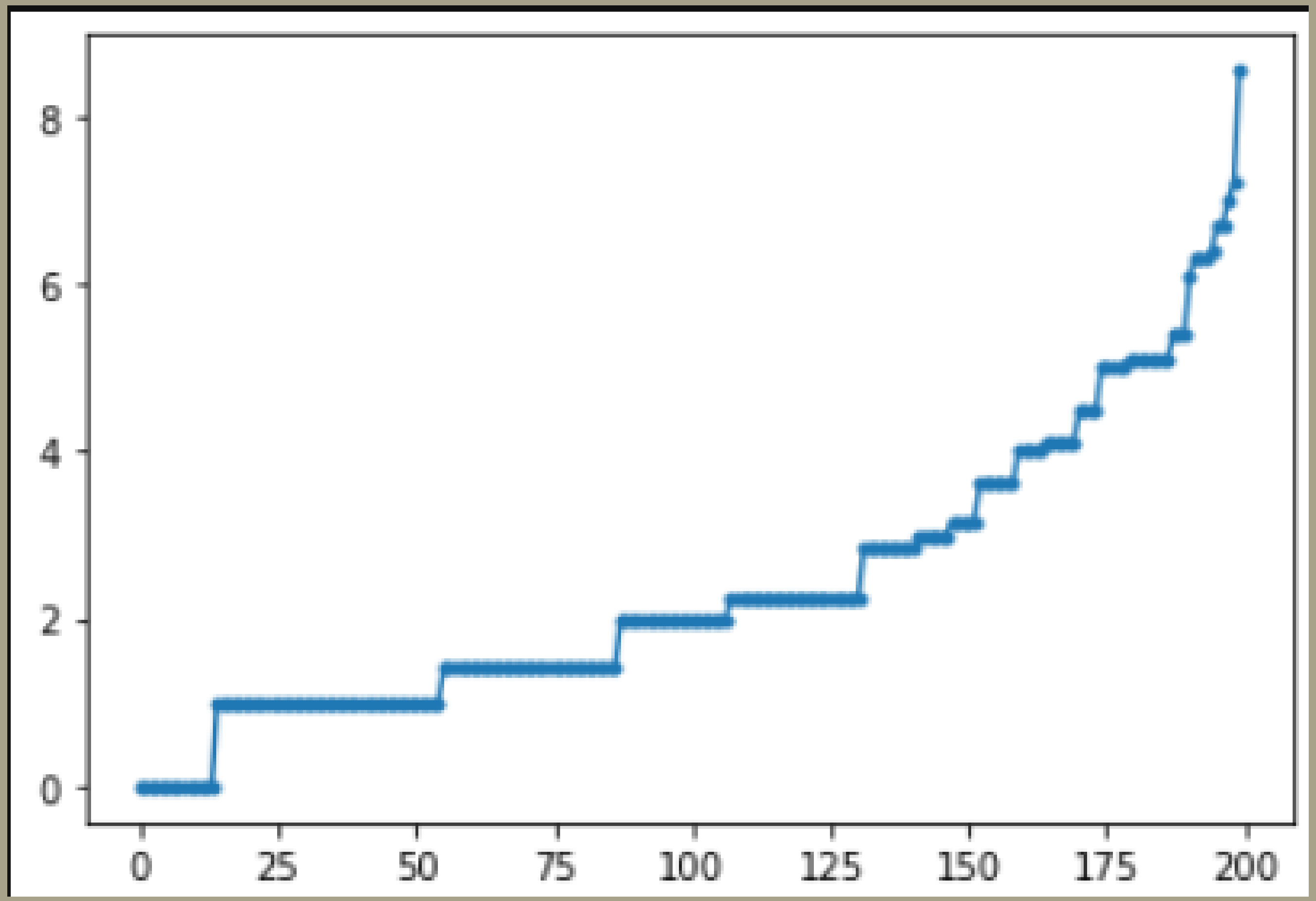
DBSCAN is a density-based clustering method from the position of data observations with the principle of grouping data that are relatively close together.

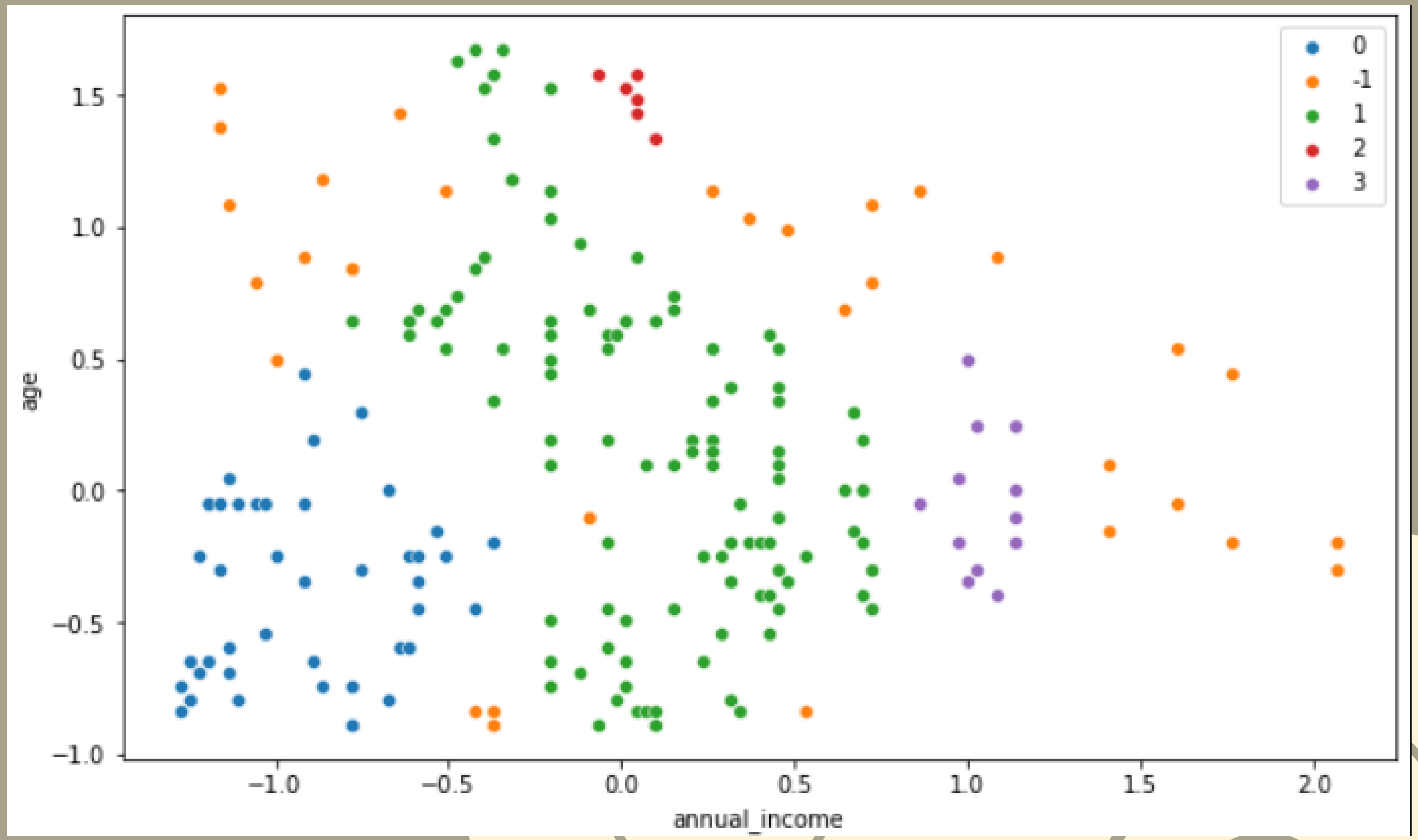


```
nb = NearestNeighbors(n_neighbors=4).fit(X)
dist, idx = nb.kneighbors(X)
```

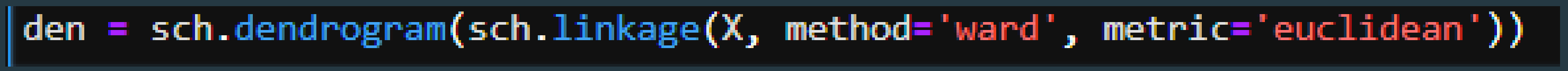


```
from sklearn.cluster import DBSCAN
model_db = DBSCAN(eps=6, min_samples=4)
model_db.fit(X)
```

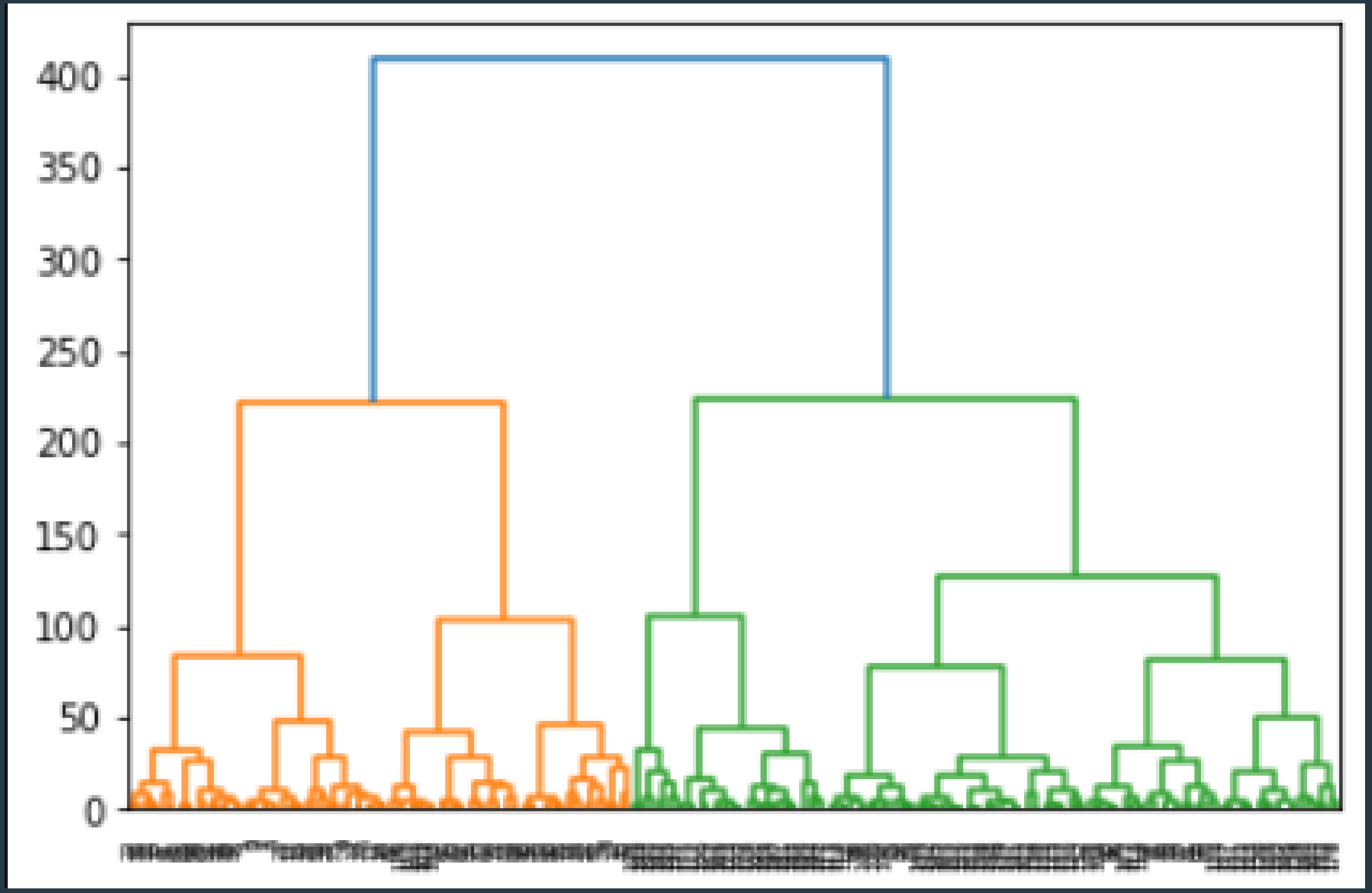




# AGGLOMERATIVE CLUSTERING



```
den = sch.dendrogram(sch.linkage(X, method='ward', metric='euclidean'))
```



Agglomerative Clustering is grouping data using a bottom-up manner, where the process will start by considering each data as a small cluster (leaf) that has one member, then two clusters that have similarities will be made into one larger cluster (node). The process will continue until it becomes one large cluster (root).

# THANK YOU!