In [1]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

In [2]:
```
model=KNeighborsClassifier()
model.partial_fit([[]],[])
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[2], line 2
      1 model=KNeighborsClassifier()
----> 2 model.partial_fit([[]],[])

AttributeError: 'KNeighborsClassifier' object has no attribute 'partial_fit'
```

In [4]:
```
model=LogisticRegression()
model.partial_fit([[]],[])
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[4], line 2
      1 model=LogisticRegression()
----> 2 model.partial_fit([[]],[])

AttributeError: 'LogisticRegression' object has no attribute 'partial_fit'
```

In [5]:
```
model=DecisionTreeClassifier()
model.partial_fit([[]],[])
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[5], line 2
      1 model=DecisionTreeClassifier()
----> 2 model.partial_fit([[]],[])

AttributeError: 'DecisionTreeClassifier' object has no attribute 'partial_fit'
```

In [6]:
```
model=SVC()
model.partial_fit([[]],[])
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[6], line 2
      1 model=SVC()
----> 2 model.partial_fit([[]],[])

AttributeError: 'SVC' object has no attribute 'partial_fit'
```

In [7]:
```
model=RandomForestClassifier()
model.partial_fit([[]],[])
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[7], line 2
      1 model=RandomForestClassifier()
----> 2 model.partial_fit([[]],[])

AttributeError: 'RandomForestClassifier' object has no attribute 'partial_fit'
```

In [8]:
```
from sklearn.linear_model import SGDClassifier
```

In [10]:
```
model=SGDClassifier()
model.partial_fit([[]],[])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[10], line 2
      1 model=SGDClassifier()
----> 2 model.partial_fit([[]],[])

File ~\anaconda3\Lib\site-packages\sklearn\linear_model\_stochastic_gradient.py:849, in
BaseSGDClassifier.partial_fit(self, X, y, classes, sample_weight)
    836     if self.class_weight == "balanced":
    837         raise ValueError(
    838             "class_weight '{0}' is not supported for "
    839             "partial_fit. In order to use 'balanced' weights,"
   (...)
    846             "parameter.".format(self.class_weight)
    847         )
--> 849 return self._partial_fit(
    850     X,
    851     y,
    852     alpha=self.alpha,
    853     C=1.0,
    854     loss=self.loss,
    855     learning_rate=self.learning_rate,
    856     max_iter=1,
    857     classes=classes,
    858     sample_weight=sample_weight,
    859     coef_init=None,
    860     intercept_init=None,
    861 )

File ~\anaconda3\Lib\site-packages\sklearn\linear_model\_stochastic_gradient.py:579, in
BaseSGDClassifier._partial_fit(self, X, y, alpha, C, loss, learning_rate, max_iter, clas
ses, sample_weight, coef_init, intercept_init)
    564 def _partial_fit(
    565     self,
    566     X,
   (...)
    576     intercept_init,
    577 ):
    578     first_call = not hasattr(self, "classes_")
--> 579     X, y = self._validate_data(
    580         X,
    581         y,
    582         accept_sparse="csr",
    583         dtype=np.float64,
    584         order="C",
    585         accept_large_sparse=False,
    586         reset=first_call,
    587     )
    589     n_samples, n_features = X.shape
    591     _check_partial_fit_first_call(self, classes)

File ~\anaconda3\Lib\site-packages\sklearn\base.py:584, in BaseEstimator._validate_data
(self, X, y, reset, validate_separately, **check_params)
    582         y = check_array(y, input_name="y", **check_y_params)
    583     else:
--> 584         X, y = check_X_y(X, y, **check_params)
    585     out = X, y
    587 if not no_val_X and check_params.get("ensure_2d", True):

File ~\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1106, in check_X_y(X, y,
 accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, al
low_nd, multi_output, ensure_min_samples, ensure_min_features, y_numeric, estimator)
```

```
   1101         estimator_name = _check_estimator_name(estimator)
   1102     raise ValueError(
   1103         f"{estimator_name} requires y to be passed, but the target y is None"
   1104     )
-> 1106 X = check_array(
   1107     X,
   1108     accept_sparse=accept_sparse,
   1109     accept_large_sparse=accept_large_sparse,
   1110     dtype=dtype,
   1111     order=order,
   1112     copy=copy,
   1113     force_all_finite=force_all_finite,
   1114     ensure_2d=ensure_2d,
   1115     allow_nd=allow_nd,
   1116     ensure_min_samples=ensure_min_samples,
   1117     ensure_min_features=ensure_min_features,
   1118     estimator=estimator,
   1119     input_name="X",
   1120 )
   1122 y = _check_y(y, multi_output=multi_output, y_numeric=y_numeric, estimator=estima
tor)
   1124 check_consistent_length(X, y)

File ~\anaconda3\Lib\site-packages\sklearn\utils\validation.py:940, in check_array(arra
y, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d,
 allow_nd, ensure_min_samples, ensure_min_features, estimator, input_name)
    938     n_features = array.shape[1]
    939     if n_features < ensure_min_features:
--> 940         raise ValueError(
    941             "Found array with %d feature(s) (shape=%s) while"
    942             " a minimum of %d is required%s."
    943             % (n_features, array.shape, ensure_min_features, context)
    944         )
    946 if copy:
    947     if xp.__name__ in {"numpy", "numpy.array_api"}:
    948         # only make a copy if `array` and `array_orig` may share memory`

ValueError: Found array with 0 feature(s) (shape=(1, 0)) while a minimum of 1 is require
d by SGDClassifier.
```

In [11]:
```
from sklearn.naive_bayes import GaussianNB
```

In [12]:
```
model=GaussianNB()
model.partial_fit([[]],[])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[12], line 2
      1 model=GaussianNB()
----> 2 model.partial_fit([[]],[])

File ~\anaconda3\Lib\site-packages\sklearn\naive_bayes.py:391, in GaussianNB.partial_fit
(self, X, y, classes, sample_weight)
    350 """Incremental fit on a batch of samples.
    351
    352 This method is expected to be called several times consecutively
   (...)
    387     Returns the instance itself.
    388 """
    389 self._validate_params()
--> 391 return self._partial_fit(
    392     X, y, classes, _refit=False, sample_weight=sample_weight
    393 )

File ~\anaconda3\Lib\site-packages\sklearn\naive_bayes.py:427, in GaussianNB._partial_fi
```

```
t(self, X, y, classes, _refit, sample_weight)
    424 if _refit:
    425     self.classes_ = None
--> 427 first_call = _check_partial_fit_first_call(self, classes)
    428 X, y = self._validate_data(X, y, reset=first_call)
    429 if sample_weight is not None:

File ~\anaconda3\Lib\site-packages\sklearn\utils\multiclass.py:408, in _check_partial_fi
t_first_call(clf, classes)
    394 """Private helper function for factorizing common classes param logic.
    395
    396 Estimators that implement the ``partial_fit`` API need to be provided with
   (...)
    405
    406 """
    407 if getattr(clf, "classes_", None) is None and classes is None:
--> 408     raise ValueError("classes must be passed on the first call to partial_fit.")
    410 elif classes is not None:
    411     if getattr(clf, "classes_", None) is not None:

ValueError: classes must be passed on the first call to partial_fit.
```

In [13]:
```
from sklearn.datasets import load_iris
```

In [14]:
```
iris=load_iris()
```

In [15]:
```
X=iris.data
y=iris.target
```

In [16]:
```
X.shape
```

Out[16]:
```
(150, 4)
```

In [18]:
```
X1=X[:100,:]
y1=y[:100]

X2=X[100:,:]
y2=y[100:]
```

In [19]:
```
X1.shape
```

Out[19]:
```
(100, 4)
```

In [20]:
```
X2.shape
```

Out[20]:
```
(50, 4)
```

In [21]:
```
model=SGDClassifier()
model.partial_fit(X1,y1,classes=[0,1,2])
```

Out[21]:
```
▾ SGDClassifier
SGDClassifier()
```

In [22]:
```
model.predict([[2.7,1.1,5.6,.2]])
```

Out[22]:
```
array([1])
```

In [23]:
```
model.partial_fit(X2,y2)
```

Out[23]:

```
▾ SGDClassifier
SGDClassifier()
```

In [24]:

```
model.predict([[2.7,1.1,5.6,.2]])
```

Out[24]:

```
array([2])
```

In [ ]: