Learn and predict.

Script.

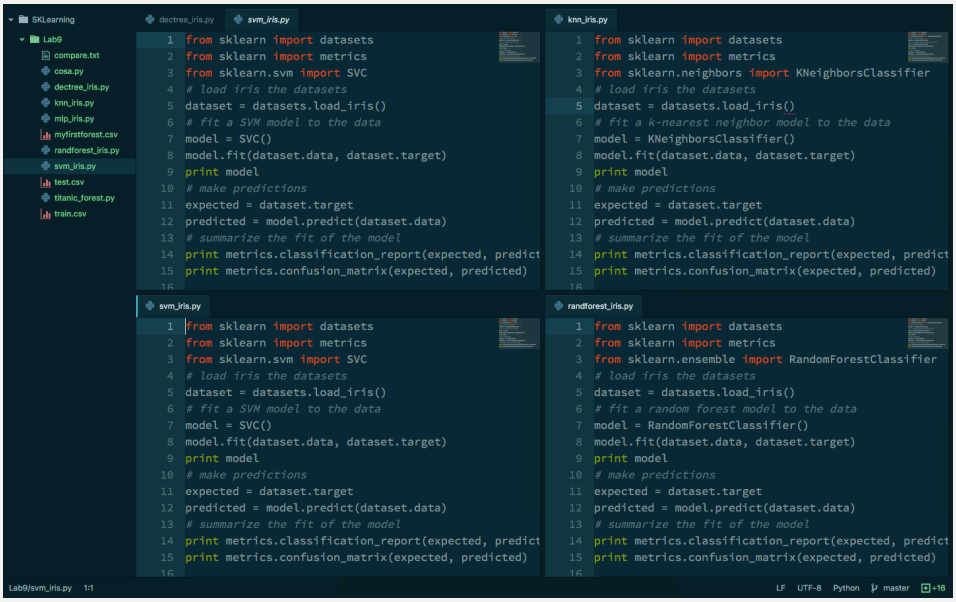
```python
import pandas as pd
from sklearn.preprocessing import Imputer
from sklearn.tree import DecisionTreeClassifier
from sklearn.cross_validation import cross_val_score

# MODEL !
RF = DecisionTreeClassifier()

# TRAIN !
trainner = pd.read_csv("train.csv")
trainner["Sex"] = trainner["Sex"].apply(lambda sex: 0 if sex == "male" else 1)

in_data = ["Fare", "Pclass", "Sex"]
bool_value = trainner["Survived"].values
data_values = trainner[list(in_data)].values

RF_score = cross_val_score(RF, data_values, bool_value).mean()

print("{0} -> ET: {1}\n".format(in_data, RF_score))

# PREDICT !

testter = pd.read_csv('test.csv')
testter["Sex"] = testter["Sex"].apply(lambda sex: 0 if sex == "male" else 1)

formatter = Imputer(missing_values='NaN', strategy='mean', axis=0)

formatter.fit(data_values)
RF.fit(data_values,bool_value)

print RF.predict(formatter.transform(testter[in_data].values))
```

Execution.

```
1. Shell
:) • SKLearning/Lab9 • master %  •
⇒  py titanic_forest.py 2> /dev/null

['Fare', 'Pclass', 'Sex'] -> ET: 0.793490460157

[0 1 0 0 1 0 0 1 1 0 0 0 1 0 1 1 0 0 1 1 0 0 1 0 1 0 1 0 1 0 0 0 1 0 1 0 0
 0 0 1 1 0 0 1 1 0 0 0 1 1 0 0 1 1 1 0 0 0 0 1 0 0 0 1 0 1 1 0 0 1 1 0 1 0
 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0
 0 1 1 1 0 0 1 1 1 1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 1 0 0 1 0 0 1 0 0 1 1 1 1 0 0 1 0 0 1 0 0 0 1 0 0 1 1 1 1 1 0 0 1 0 1
 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0
 1 0 1 0 0 1 0 0 0 1 0 0 0 0 1 0 1 1 1 1 1 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1
 0 0 0 1 1 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0
 1 0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0
 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 1 0 0 0 1 0 1 0 0 1 0 1 1 0 1 0 0 0 1 1
 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 1 0 0 0 1 0 1 0 0 1 0 1 0 0 1 0
 0 1 1 1 1 0 0 1 0 0 0]

:) • SKLearning/Lab9 • master %  •
⇒
```

Comparison.

Snippets.



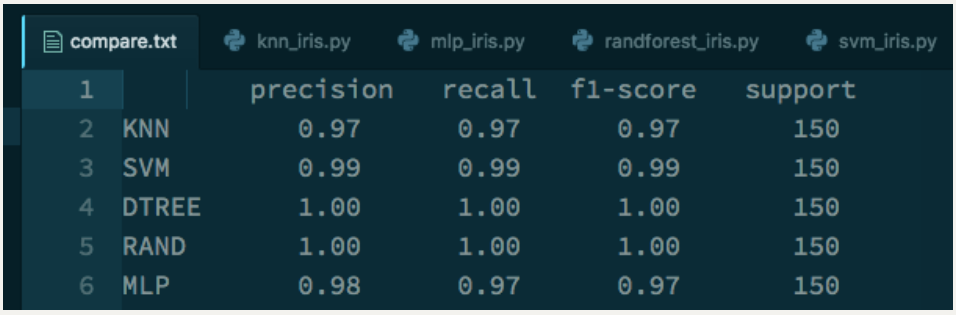

```
from sklearn import datasets
from sklearn import metrics
from sklearn.svm import SVC
# load iris the datasets
dataset = datasets.load_iris()
# fit a SVM model to the data
model = SVC()
model.fit(dataset.data, dataset.target)
print model
# make predictions
expected = dataset.target
predicted = model.predict(dataset.data)
# summarize the fit of the model
print metrics.classification_report(expected, predict
print metrics.confusion_matrix(expected, predicted)
```

```
from sklearn import datasets
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
# load iris the datasets
dataset = datasets.load_iris()
# fit a k-nearest neighbor model to the data
model = KNeighborsClassifier()
model.fit(dataset.data, dataset.target)
print model
# make predictions
expected = dataset.target
predicted = model.predict(dataset.data)
# summarize the fit of the model
print metrics.classification_report(expected, predict
print metrics.confusion_matrix(expected, predicted)
```

```
from sklearn import datasets
from sklearn import metrics
from sklearn.svm import SVC
# load iris the datasets
dataset = datasets.load_iris()
# fit a SVM model to the data
model = SVC()
model.fit(dataset.data, dataset.target)
print model
# make predictions
expected = dataset.target
predicted = model.predict(dataset.data)
# summarize the fit of the model
print metrics.classification_report(expected, predict
print metrics.confusion_matrix(expected, predicted)
```

```
from sklearn import datasets
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
# load iris the datasets
dataset = datasets.load_iris()
# fit a random forest model to the data
model = RandomForestClassifier()
model.fit(dataset.data, dataset.target)
print model
# make predictions
expected = dataset.target
predicted = model.predict(dataset.data)
# summarize the fit of the model
print metrics.classification_report(expected, predict
print metrics.confusion_matrix(expected, predicted)
```

Table.



	precision	recall	f1-score	support
KNN	0.97	0.97	0.97	150
SVM	0.99	0.99	0.99	150
DTREE	1.00	1.00	1.00	150
RAND	1.00	1.00	1.00	150
MLP	0.98	0.97	0.97	150