Predictive modeling ~= machine learning

- Make predictions of outcome on new data
- Extract the structure of historical data
- Statistical tools to summarize the training data into a executable predictive model
- Alternative to hard-coded rules written by experts

type (category)	# rooms (int)	surface (float m2)	public trans (boolean)
Apartment	3	50	TRUE
House	5	254	FALSE
Duplex	4	68	TRUE
Apartment	2	32	TRUE

type (category)	# rooms (int)	surface (float m2)	public trans (boolean)	sold (float k€)
Apartment	3	50	TRUE	450
House	5	254	FALSE	430
Duplex	4	68	TRUE	712
Apartment	2	32	TRUE	234

features

target

samples (train)

type (category)	# rooms (int)	surface (float m2)	public trans (boolean)	sold (float k€)
Apartment	3	50	TRUE	450
House	5	254	FALSE	430
Duplex	4	68	TRUE	712
Apartment	2	32	TRUE	234

features

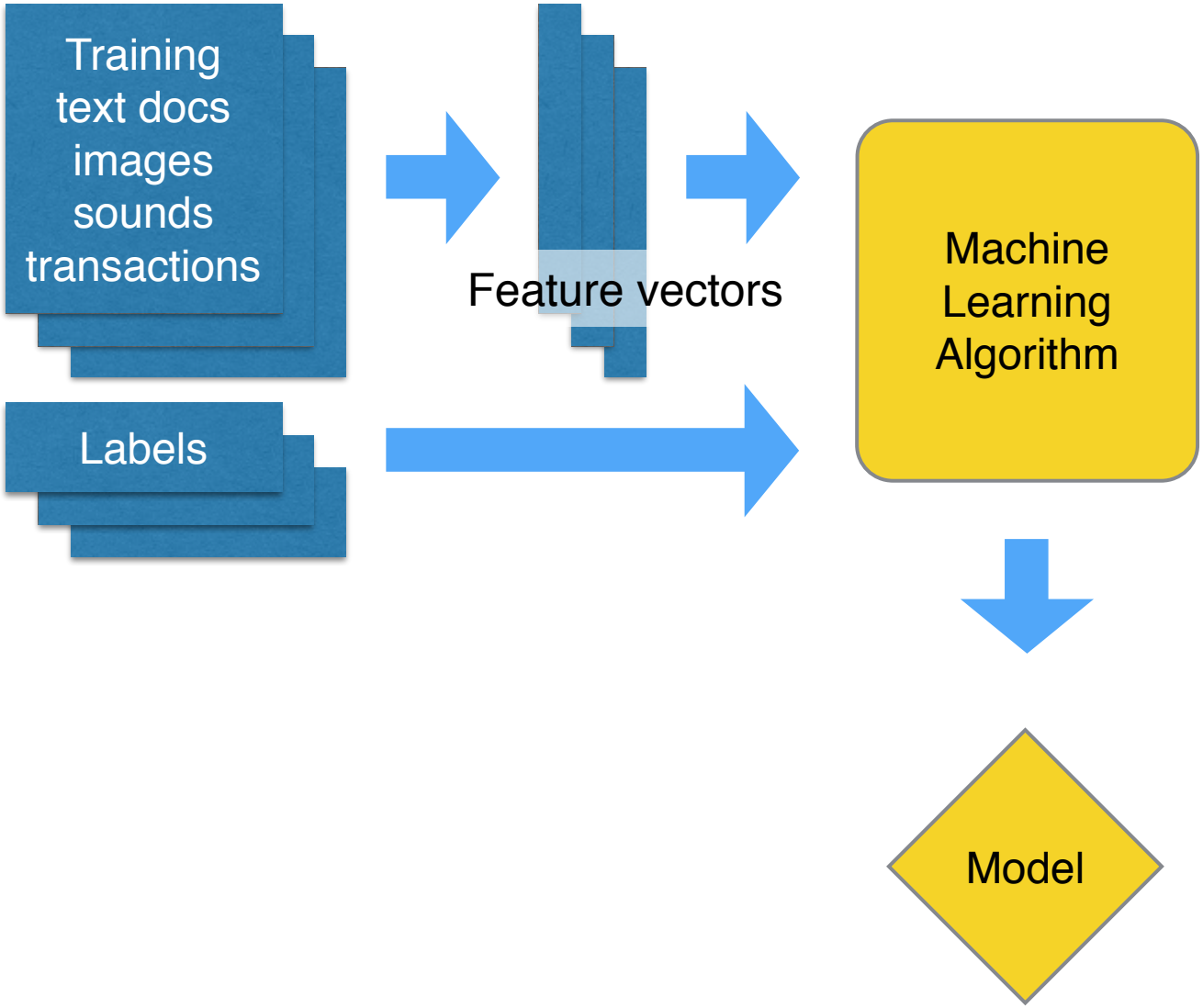
target

samples (train)

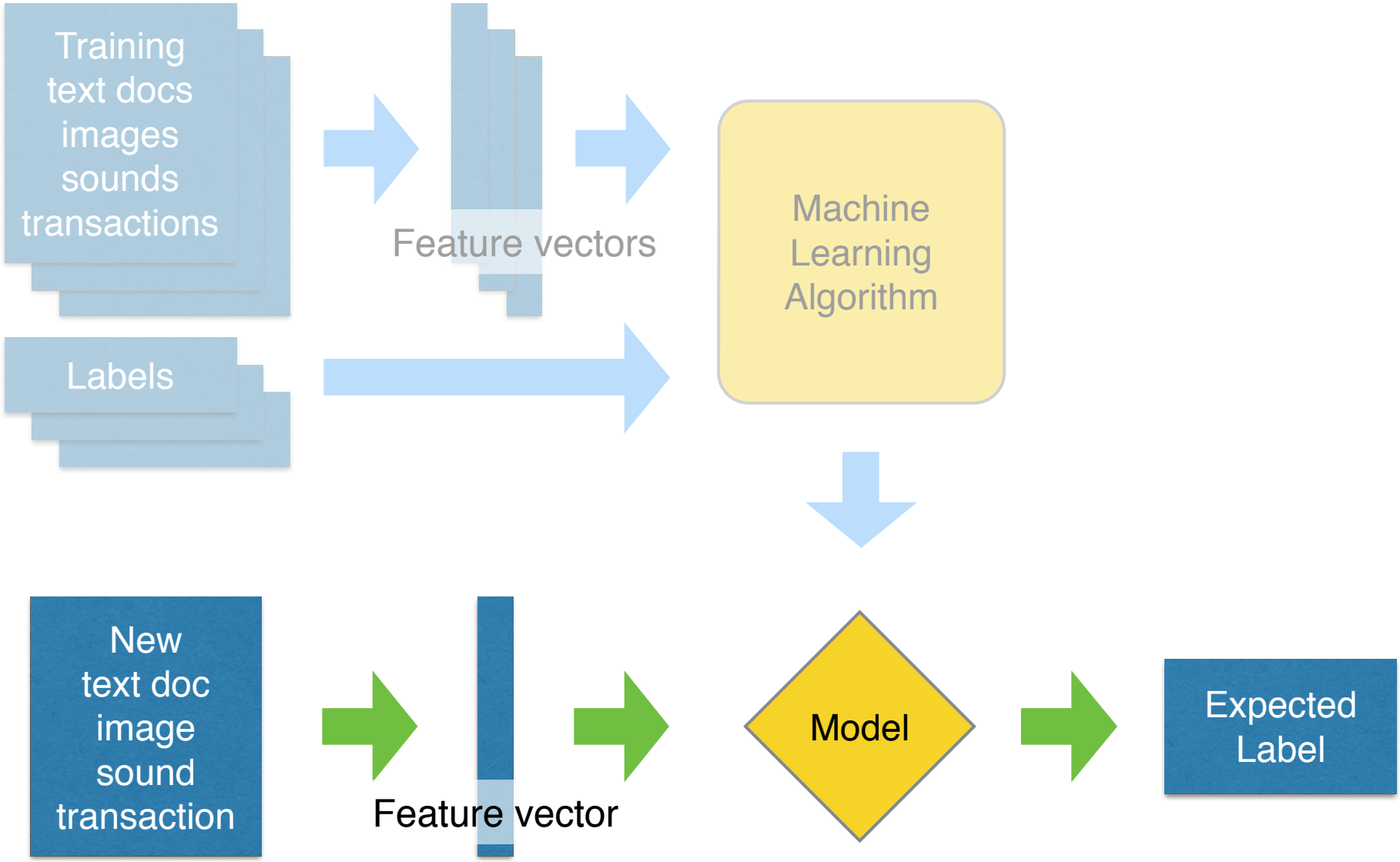
type (category)	# rooms (int)	surface (float m2)	public trans (boolean)	sold (float k€)
Apartment	3	50	TRUE	450
House	5	254	FALSE	430
Duplex	4	68	TRUE	712
Apartment	2	32	TRUE	234

samples (test)

Apartment	2	33	TRUE	?
House	4	210	TRUE	?



Predictive Modeling Data Flow



Predictive Modeling Data Flow

Predictive modeling in the wild

Virality and readers engagement

Fraud detection

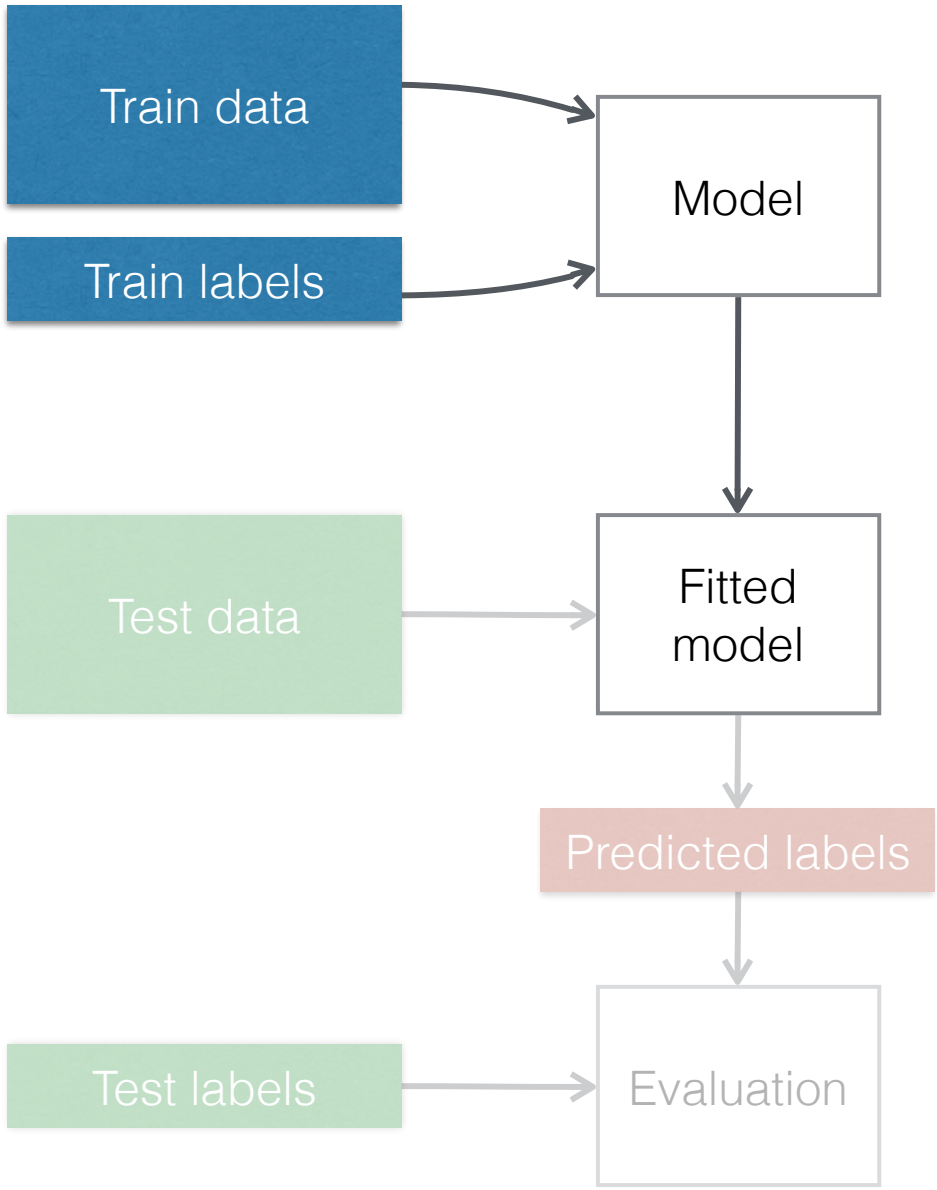
Personalized radios

Inventory forecasting & trends detection

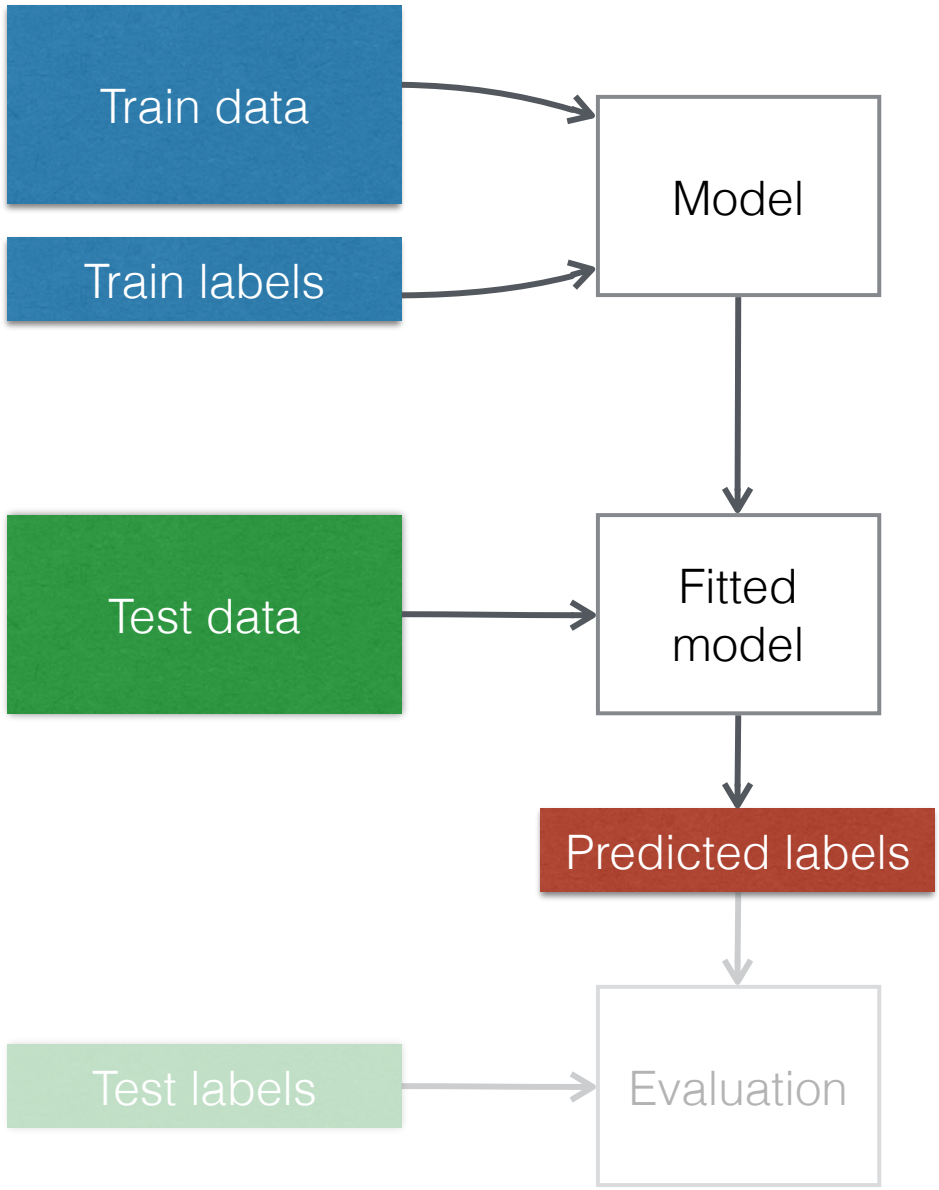
Predictive maintenance

Personality matching

- Library of Machine Learning algorithms
- Focus on established methods (e.g. ESL-II)
- Open Source (BSD)
- Simple **`fit`** / **`predict`** / **`transform`** API
- Python / NumPy / SciPy / Cython
- Model Assessment, Selection & Ensembles

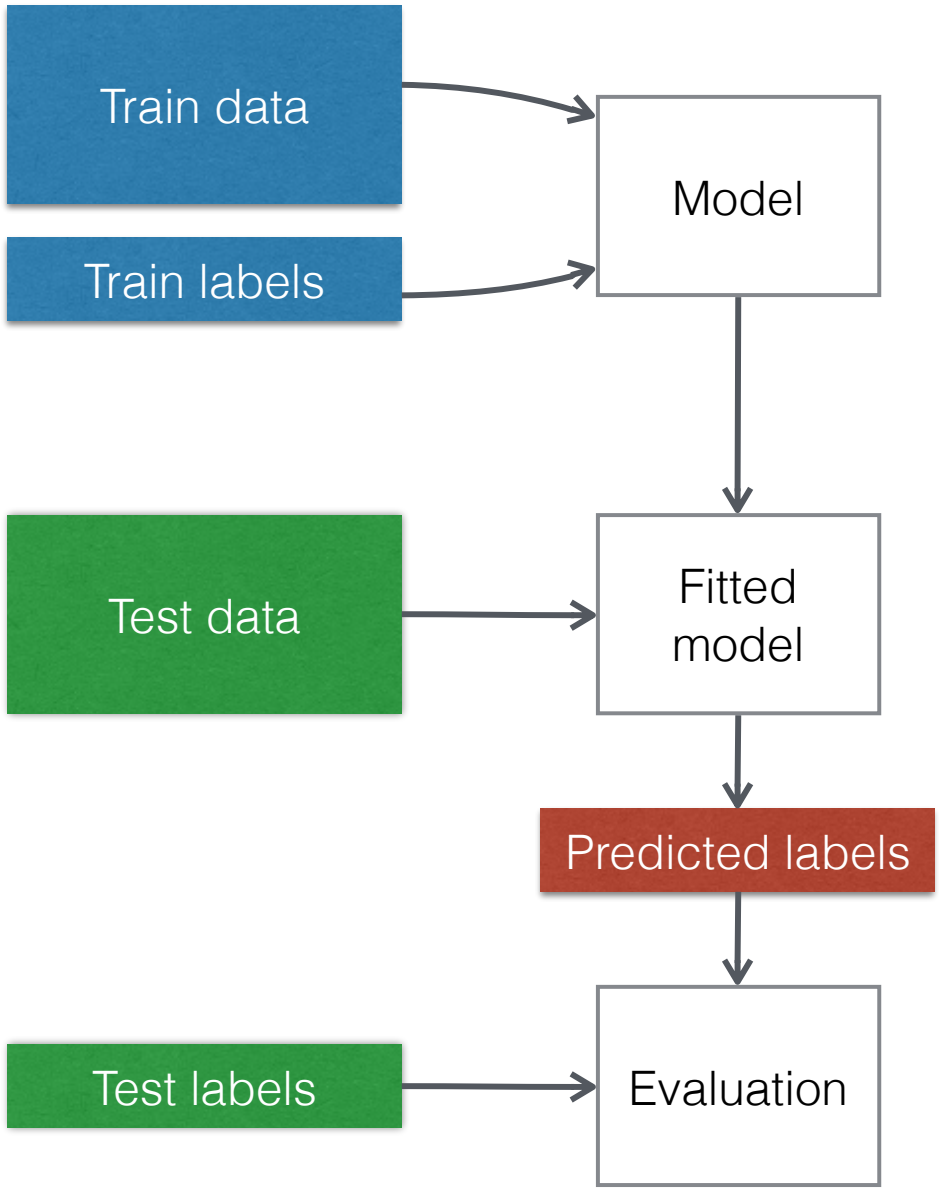



```
model = ModelClass(**hyperparams)
model.fit(X_train, y_train)
```




```
model = ModelClass(**hyperparams)
model.fit(X_train, y_train)
```

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




```
model = ModelClass(**hyperparams)
model.fit(X_train, y_train)
```

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

```
accuracy_score(y_test, y_pred)
```

Support Vector Machine

```
from sklearn.svm import SVC

model = SVC(kernel="rbf", C=1.0, gamma=1e-4)

model.fit(X_train, y_train)


y_predicted = model.predict(X_test)

from sklearn.metrics import f1_score
f1_score(y_test, y_predicted)
```

Linear Classifier

```
from sklearn.linear_model import SGDClassifier

model = SGDClassifier(alpha=1e-4,
                      penalty="elasticnet")
model.fit(X_train, y_train)


y_predicted = model.predict(X_test)

from sklearn.metrics import f1_score
f1_score(y_test, y_predicted)
```

Random Forests

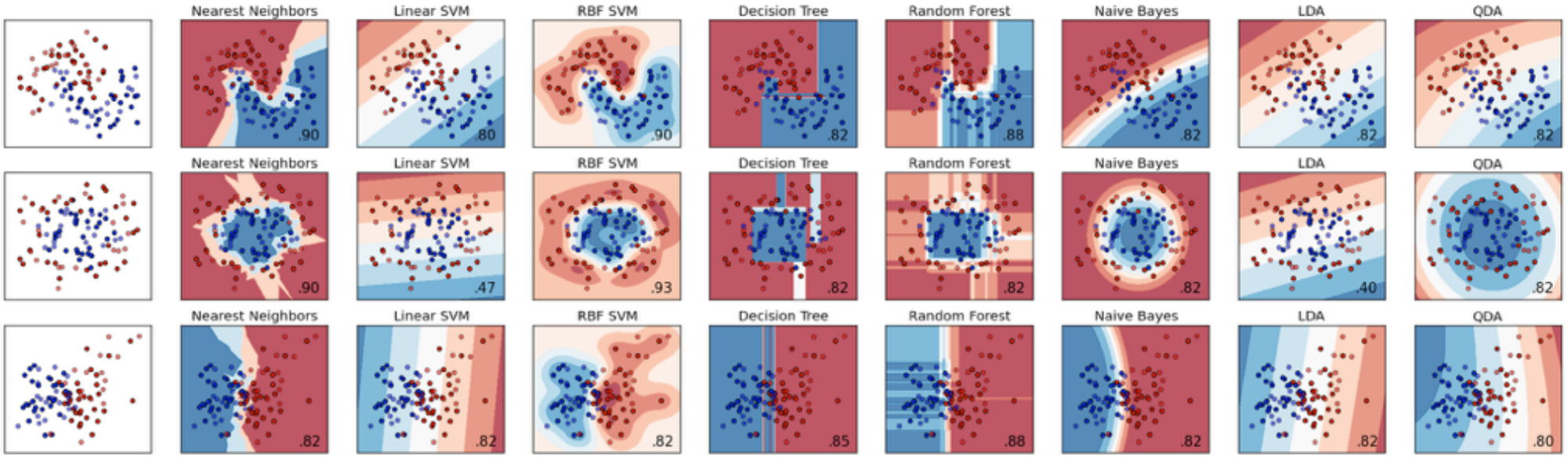
```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=200)

model.fit(X_train, y_train)


y_predicted = model.predict(X_test)

from sklearn.metrics import f1_score
f1_score(y_test, y_predicted)
```


Nearest Neighbors
Linear SVM
RBF SVM
Decision Tree
Random Forest
Naive Bayes
LDA
QDA
.90
.80
.90
.82
.88
.82
.82
.82
Nearest Neighbors
Linear SVM
RBF SVM
Decision Tree
Random Forest
Naive Bayes
LDA
QDA
.90
.47
.93
.82
.82
.82
.40
.82
Nearest Neighbors
Linear SVM
RBF SVM
Decision Tree
Random Forest
Naive Bayes
LDA
QDA
.82
.82
.82
.85
.88
.82
.82
.80