

Intro to Scikit-Learn

Pablo Maldonado

Representing Data

one sample

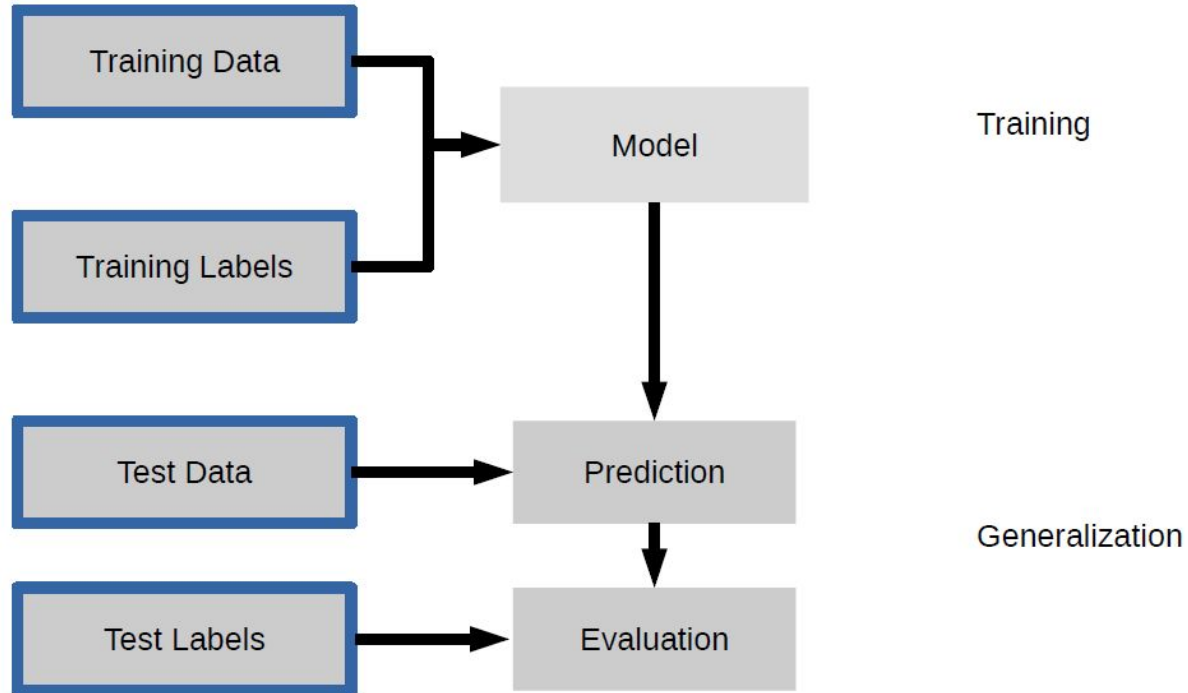
$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix}$$

one feature

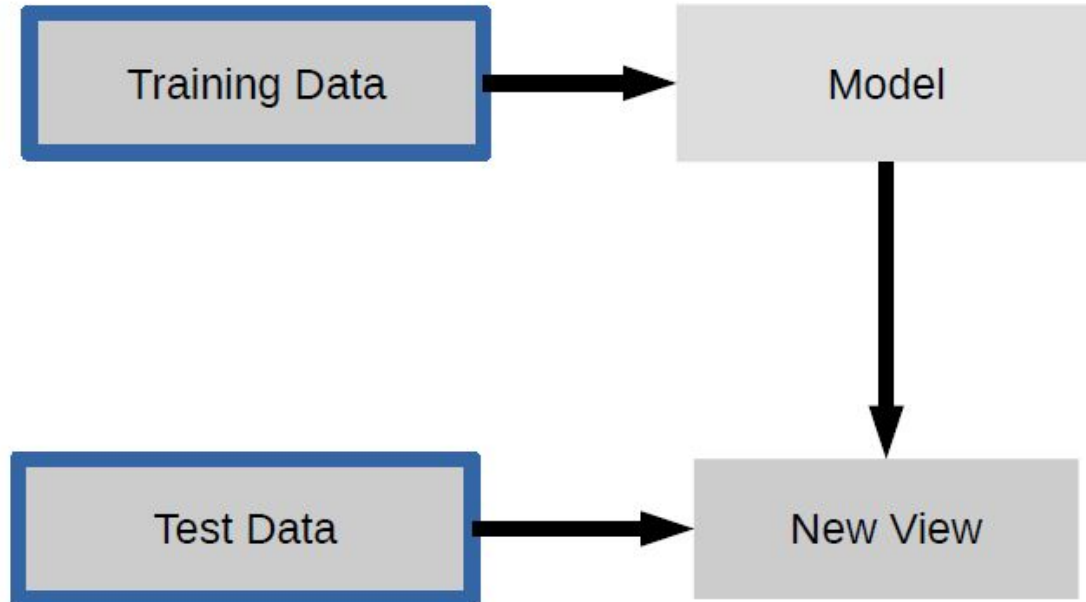
$$y = \begin{pmatrix} 1.6 \\ 2.7 \\ 4.4 \\ 0.5 \\ 0.2 \\ 5.6 \\ 6.7 \end{pmatrix}$$

outputs / labels

Supervised Learning



Unsupervised Learning



Examples?

The sklearn API

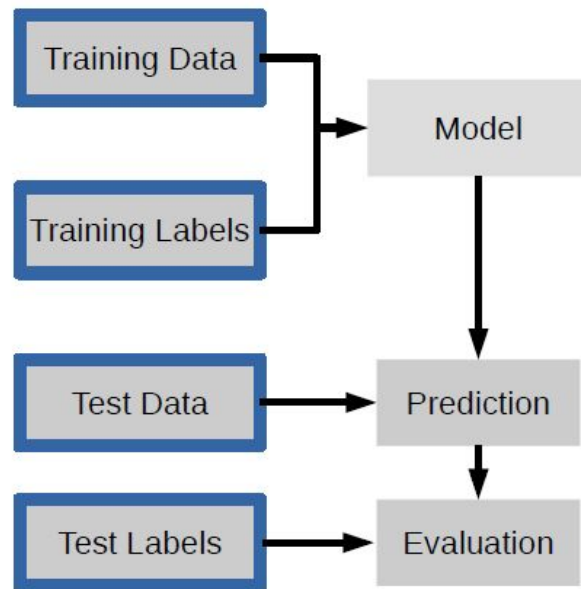
Supervised Learning

```
clf = RandomForestClassifier()
```

```
clf.fit(X_train, y_train)
```

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

```
clf.score(X_test, y_test)
```

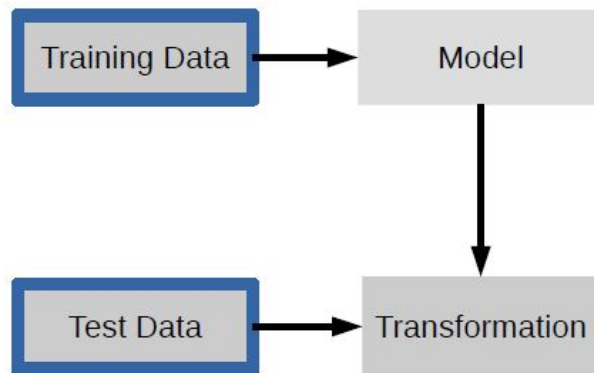


Unsupervised Learning / Transformations

```
pca = PCA()
```

```
pca.fit(X_train)
```

```
X_new = pca.transform(X_test)
```



TL;DR

`estimator.fit(X, [y])`

`estimator.predict`

`estimator.transform`

Classification

Preprocessing

Regression

Dimensionality reduction

Clustering

Feature selection

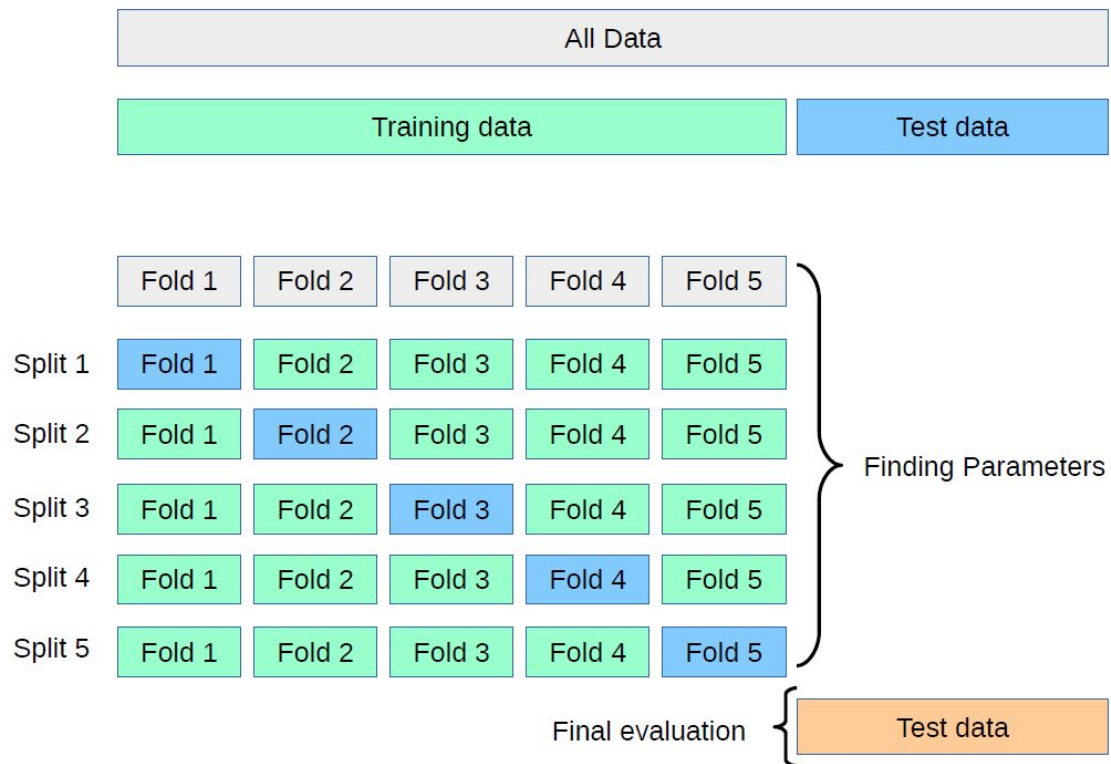
Feature extraction

Model Selection

Cross-validation



Model Validation



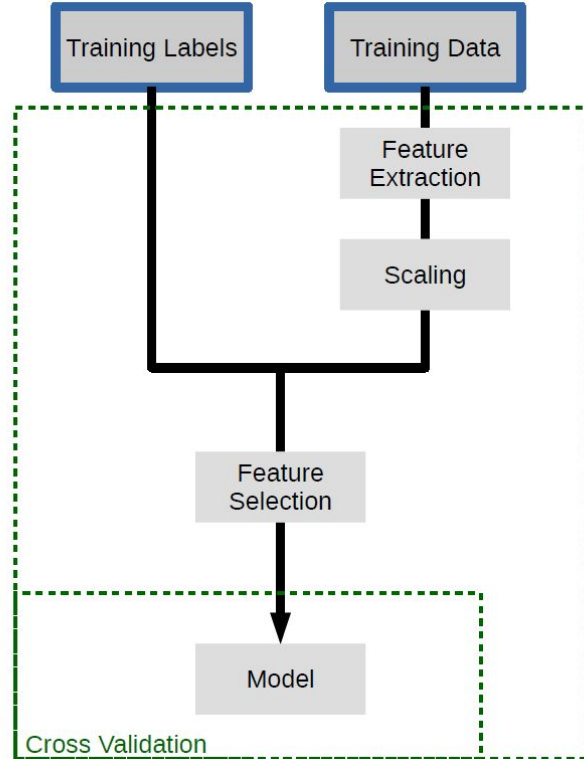
Grid Search Cross-Validation

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y)

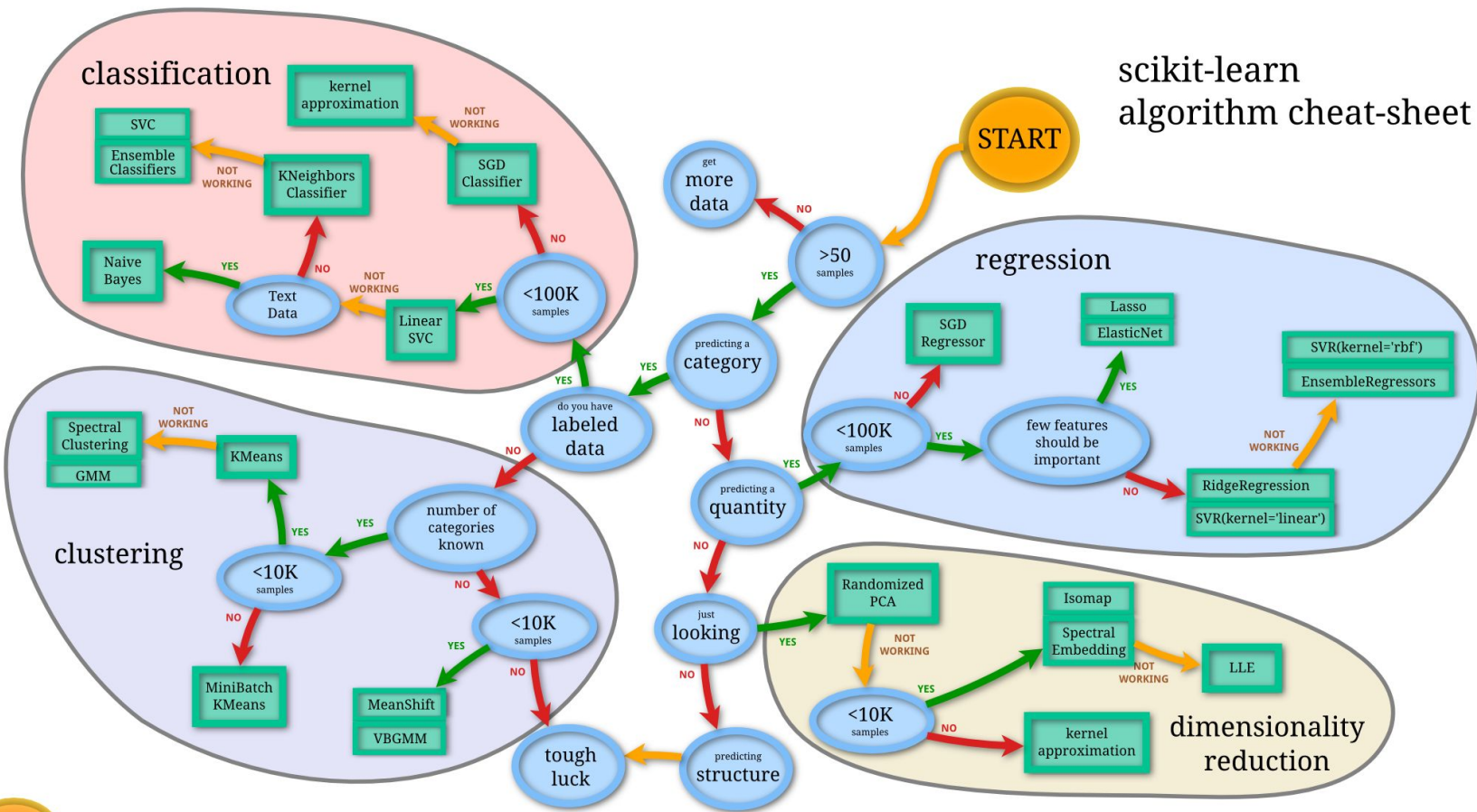
param_grid = {'C': 10. ** np.arange(-3, 3),
              'gamma': 10. ** np.arange(-3, 3)}
grid = GridSearchCV(SVC(), param_grid=param_grid)
grid.fit(X_train, y_train)
grid.predict(X_test)
grid.score(X_test, y_test)
```

Workflow

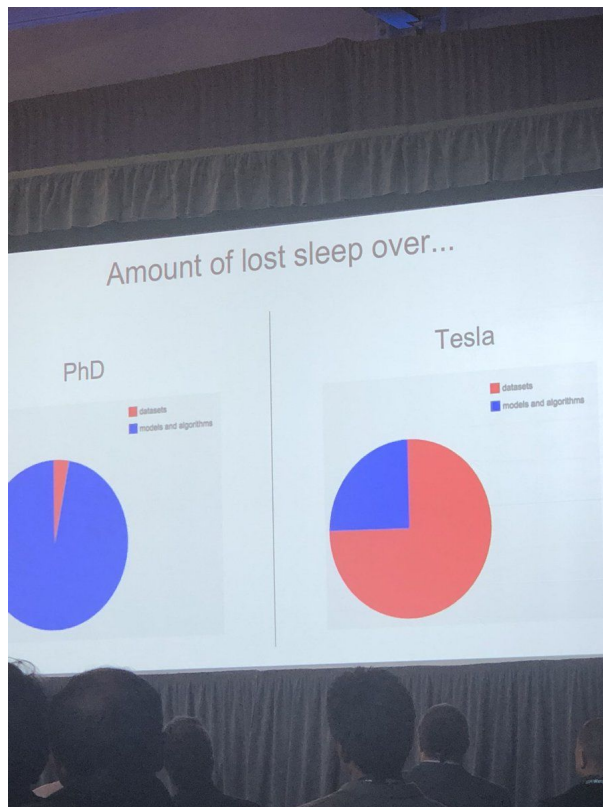


Which model to
choose?

scikit-learn
algorithm cheat-sheet



Disclaimer

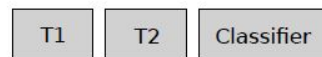


*Andrej Karpathy,
Director of AI at Tesla*

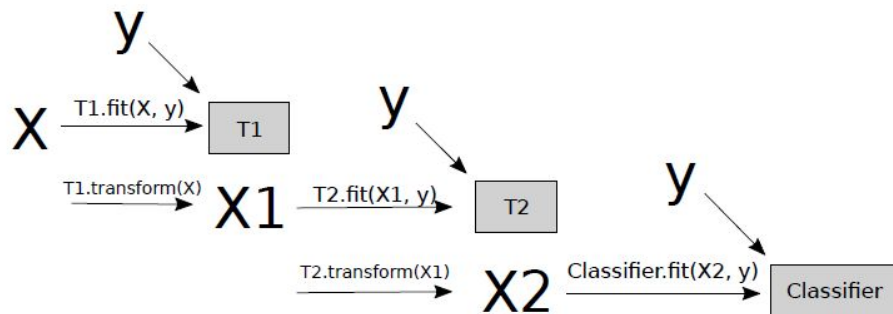
Pipelines

Pipelines

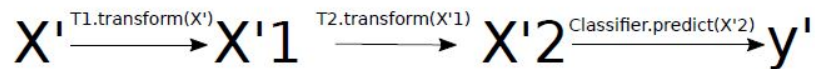
```
pipe = make_pipeline(T1(), T2(), Classifier())
```



```
pipe.fit(X, y)
```



```
pipe.predict(X')
```



```
from sklearn.pipeline import make_pipeline
```

```
pipe = make_pipeline(StandardScaler(), SVC())  
pipe.fit(X_train, y_train)  
pipe.predict(X_test)
```

```
param_grid = {'svc__C': 10. ** np.arange(-3, 3),  
              'svc__gamma': 10. ** np.arange(-3, 3)}
```

```
scaler_pipe = make_pipeline(StandardScaler(), SVC())  
grid = GridSearchCV(scaler_pipe, param_grid=param_grid, cv=5)  
grid.fit(X_train, y_train)
```

Hyperparameter search

```
param_grid = {'svc__C': 10. ** np.arange(-3, 3),  
              'svc__gamma': 10. ** np.arange(-3, 3)}  
  
scaler_pipe = make_pipeline(StandardScaler(), SVC())  
grid = GridSearchCV(scaler_pipe, param_grid=param_grid, cv=5)  
grid.fit(X_train, y_train)
```

Hyperparameter search for feature selection

```
param_grid = {'selectkbest__k': [1, 2, 3, 4],  
              'svc__C': 10. ** np.arange(-3, 3),  
              'svc__gamma': 10. ** np.arange(-3, 3)}  
  
scaler_pipe = make_pipeline(SelectKBest(), SVC())  
grid = GridSearchCV(scaler_pipe, param_grid=param_grid, cv=5)  
grid.fit(X_train, y_train)
```