Practical Machine Learning with Scikit-Learn

Lecture 11 – HCCDA-AI

Imran Nawar

12 July 2025

Introduction to Scikit-Learn

What is Scikit-Learn?

- A free, open-source Python library for machine learning.
- Built on top of NumPy, SciPy, and Matplotlib.
- Developed initially by David Cournapeau during Google Summer of Code (2007).
- Licensed under BSD, widely adopted and actively maintained.

- **Installation and Setup:**

```
pip install scikit-learn
```

```
import sklearn
```

```
(DIP_7th) PS C:\Users\PC> pip install scikit-learn
Collecting scikit-learn
  Downloading scikit_learn-1.6.1-cp312-cp312-win_amd64.whl.
metadata (15 kB)
Collecting numpy>=1.19.5 (from scikit-learn)
  Downloading numpy-2.2.2-cp312-cp312-win_amd64.whl.metadat
a (60 kB)
Collecting scipy>=1.6.0 (from scikit-learn)
  Downloading scipy-1.15.1-cp312-cp312-win_amd64.whl.metada
ta (60 kB)
Collecting joblib>=1.2.0 (from scikit-learn)
  Using cached joblib-1.4.2-py3-none-any.whl.metadata (5.4
kB)
Collecting threadpoolctl>=3.1.0 (from scikit-learn)
  Using cached threadpoolctl-3.5.0-py3-none-any.whl.metadat
a (13 kB)
Downloading scikit_learn-1.6.1-cp312-cp312-win_amd64.whl (1
1.1 MB)
   ---------------------------------------- 11.1/11.1 MB 2.
3 MB/s eta 0:00:00
Using cached joblib-1.4.2-py3-none-any.whl (301 kB)
Downloading numpy-2.2.2-cp312-cp312-win_amd64.whl (12.6 MB)
   ---------------------------------------- 12.6/12.6 MB 2.
3 MB/s eta 0:00:00
Downloading scipy-1.15.1-cp312-cp312-win_amd64.whl (43.6 MB
)
   ---------------------------------------- 43.6/43.6 MB 2.
1 MB/s eta 0:00:00
Using cached threadpoolctl-3.5.0-py3-none-any.whl (18 kB)
Installing collected packages: threadpoolctl, numpy, joblib
, scipy, scikit-learn
Successfully installed joblib-1.4.2 numpy-2.2.2 scikit-lear
n-1.6.1 scipy-1.15.1 threadpoolctl-3.5.0
```

Introduction to Scikit-Learn

What can it do?

- Scikit-Learn supports a wide range of machine learning tasks:
  - **Supervised learning**
    - Regression, Classification
  - **Unsupervised learning**
    - Clustering, Dimensionality Reduction
  - **Data Preprocessing**
    - Cleaning, encoding, scaling, and feature selection.

Why use Scikit-Learn?

- **Ease of Use:** Consistent API design makes it beginner-friendly.
- **Comprehensive Documentation:** Extensive tutorials and examples.
- Strong community and ecosystem integration (NumPy, Pandas, Matplotlib)

Data Preprocessing with Scikit Learn

Handling Missing Data:

- Use `SimpleImputer` to fill missing values.

```
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy='mean')
X_train = imputer.fit_transform(X_train)
```

Input data

1.0	January	1.5
1.4	February	0.3
5.0	March	1

→

Features matrix

1.0	0	1.5
1.4	1	0.3
5.0	2	1

Encoding Categorical Variables:

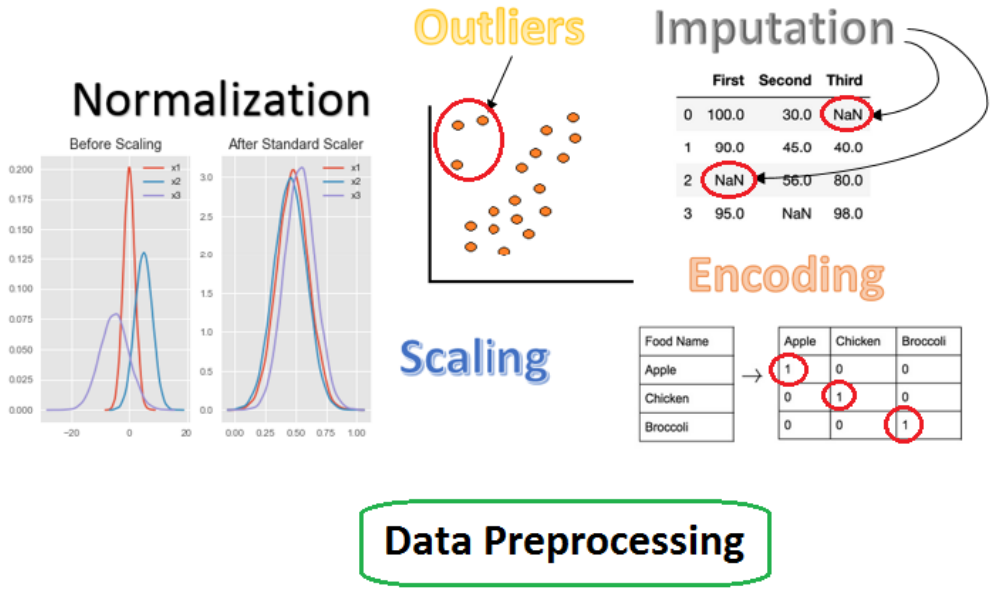
- Use `OneHotEncoder` for nominal data and `LabelEncoder` for ordinal data.

Feature Scaling:

- `StandardScaler`: Standardize features.
- `MinMaxScaler`: Normalize to a fixed range.

Feature Extraction / Dimensionality Reduction:

- Use PCA for dimensionality reduction.



Building Machine Learning Models

Scikit learn models follow a simple, shared pattern:

Step	Description
1. Import Model	Choose a model from `sklearn.linear_model`, `sklearn.tree`, etc.
2. Initialize Model	Set hyperparameters when creating the model object
3. Split Data	Use `train_test_split()` to split your dataset
4. Train (fit)	Learn model parameters using training data
5. Predict (predict)	Generate predictions on unseen (test) data

Building Machine Learning Models

Example: Linear Regression

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = LinearRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

Model Evaluation Metrics

- Evaluation metrics are used to assess the performance of machine learning models.
- Different tasks require different metrics, depending on the type of prediction problem.
- **For Classification Tasks:**
  - **Accuracy:**
    - The proportion of correct predictions out of all predictions made.

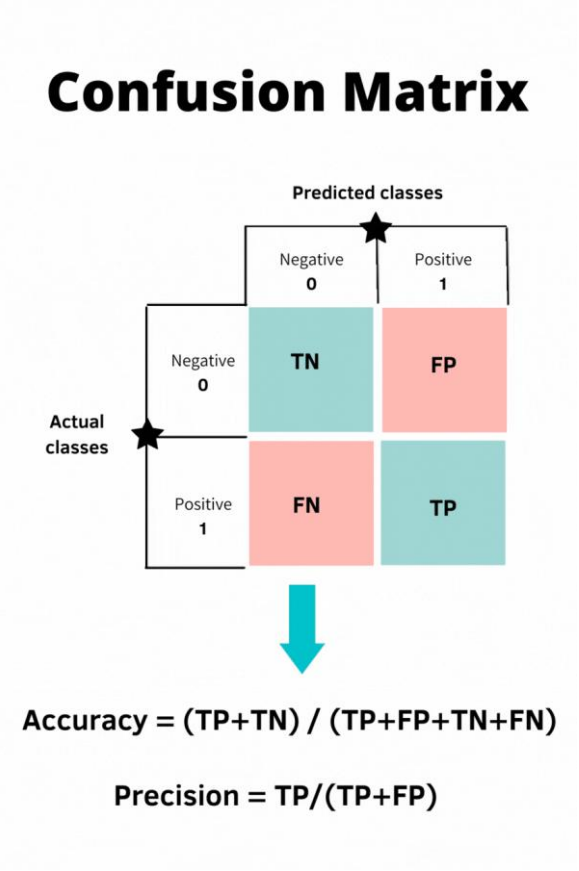
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_true, y_pred)
```

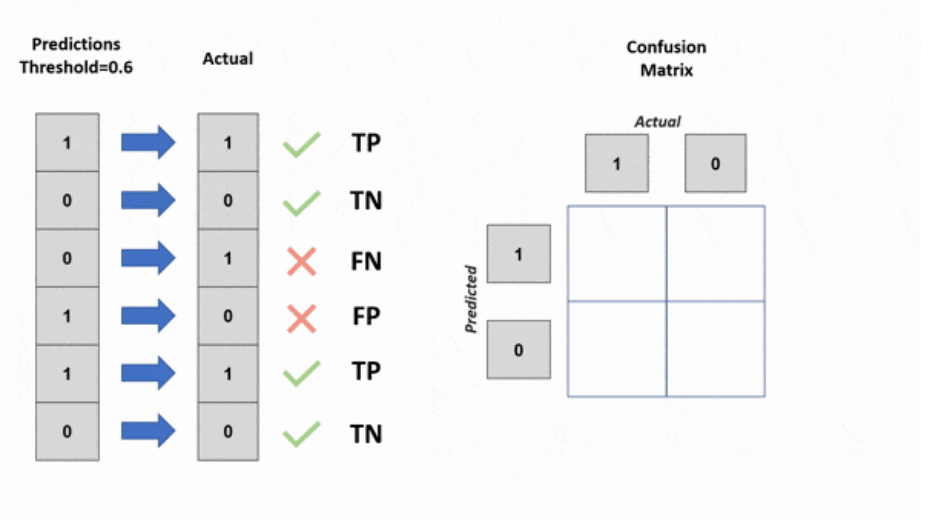
  - **Precision, Recall, F1-Score:** For imbalanced datasets.

```
from sklearn.metrics import precision_score, recall_score, f1_score
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)
f1 = f1_score(y_true, y_pred)
```

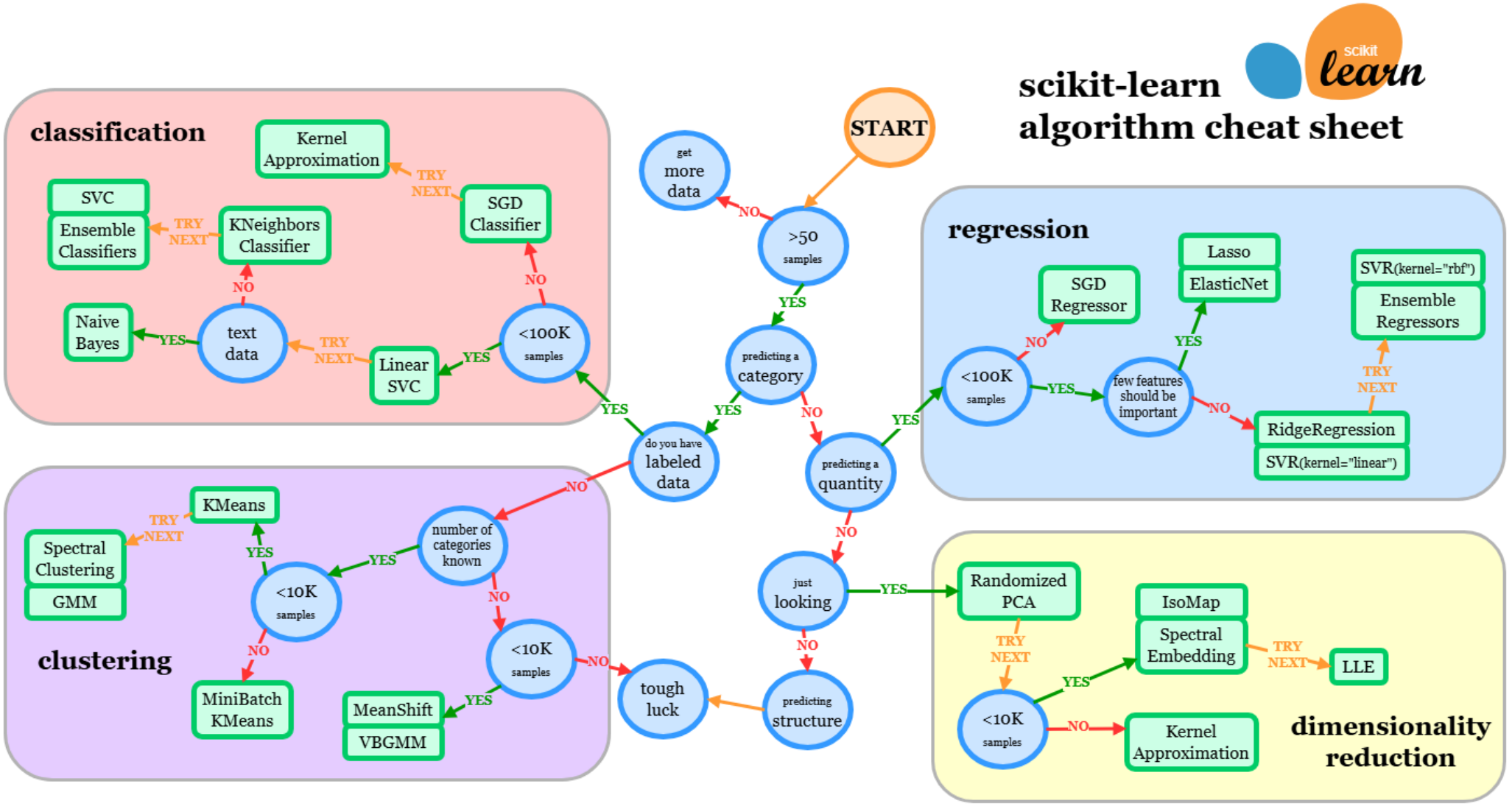
  - **Confusion Matrix:**
    - Visual representation of model performance.
    - Showing true positives, false positives, true negatives, and false negatives.

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_true, y_pred)
```





Scikit-learn Algorithm Cheat Sheet



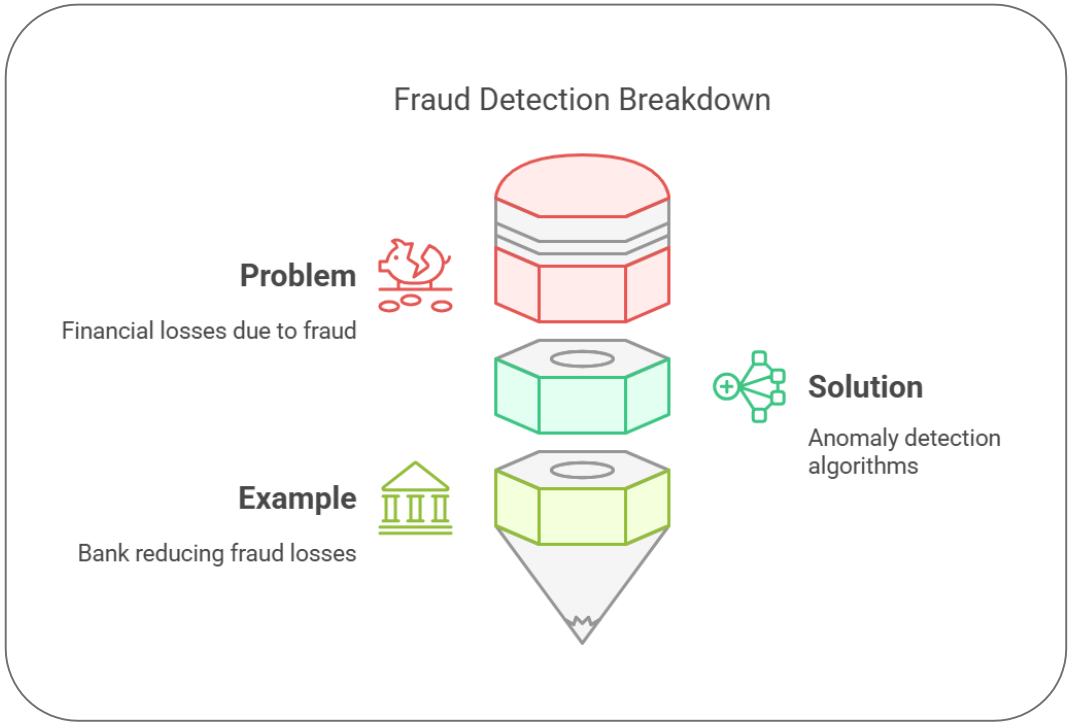
Practical Applications of Scikit-Learn

1. **Predictive Maintenance (Preventing Equipment Failure)**
   - Manufacturing industries often face costly delays due to unexpected machine breakdowns.
   - By installing sensors that collect data like temperature and vibration, companies can monitor equipment health in real-time.
   - Using Scikit-Learn's **Random Forest** algorithm, one manufacturing plant trained a model to predict failures in advance. This allowed them to schedule maintenance proactively, reducing downtime by **30%** with a prediction accuracy of **95%**.

2. **Customer Segmentation (Smarter Marketing Strategies)**
   - To better target their marketing efforts, an e-commerce business analyzed customer purchasing behavior. With the help of **K-Means clustering** from Scikit-Learn, they grouped customers into meaningful segments – such as **frequent buyers**, **seasonal shoppers**, and **discount seekers**.
   - This segmentation helped tailor promotions and messages, ultimately increasing marketing campaign effectiveness by **20%**.

Practical Applications of Scikit-Learn

3. **Fraud Detection (Catching Suspicious Transactions)**
   - Financial institutions are constantly under threat from fraudulent transactions. One bank used Scikit-Learn's **Isolation Forest** algorithm to detect outliers in customer spending patterns.
   - By flagging suspicious activity in real time, the bank was able to act faster and reduce fraud-related losses by **40%**.



Recent Advancements and Future Directions

- **Integration with Deep Learning Frameworks:**
  - Scikit-Learn can be used alongside TensorFlow and PyTorch.
  - **Example**: Use Scikit-Learn for preprocessing data before feeding it into a TensorFlow neural network.

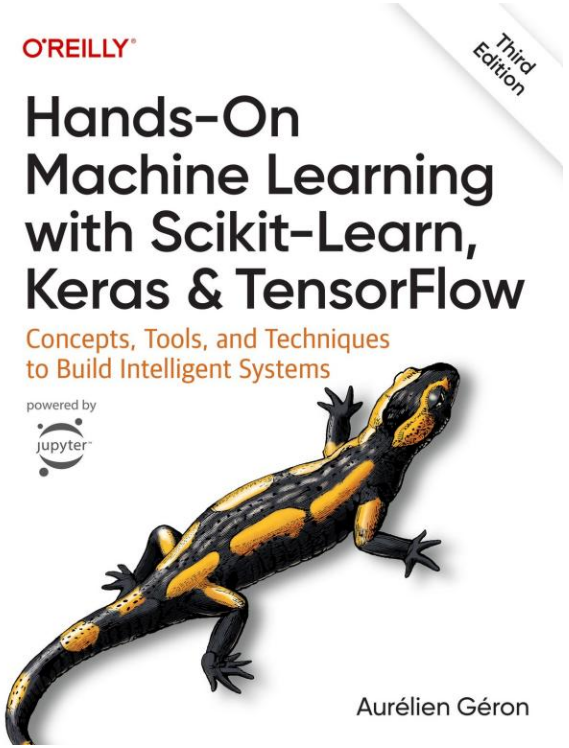
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

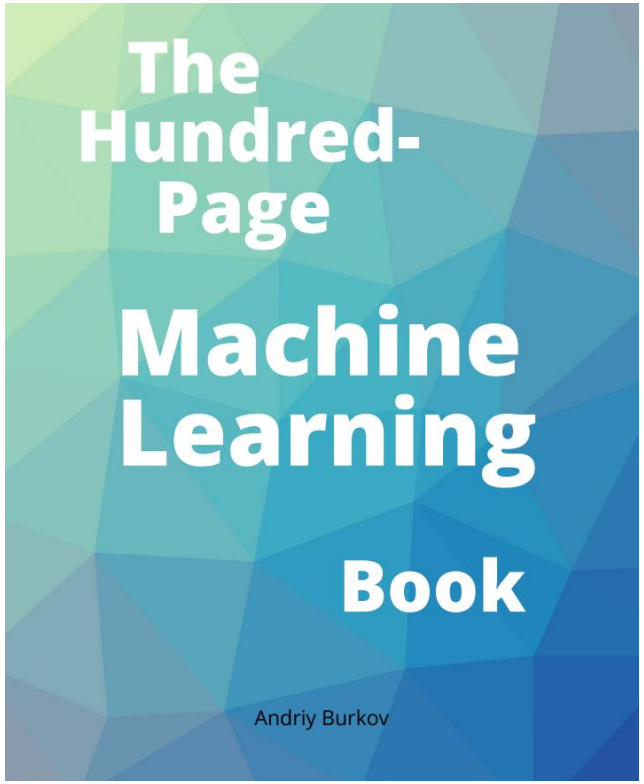
- **Automated Machine Learning (AutoML):**
  - Tools like **Auto-Sklearn** automate model selection, hyperparameter tuning, and feature engineering.
  - **Example:** Auto-Sklearn can automatically select the best model for a dataset, saving hours of manual effort.

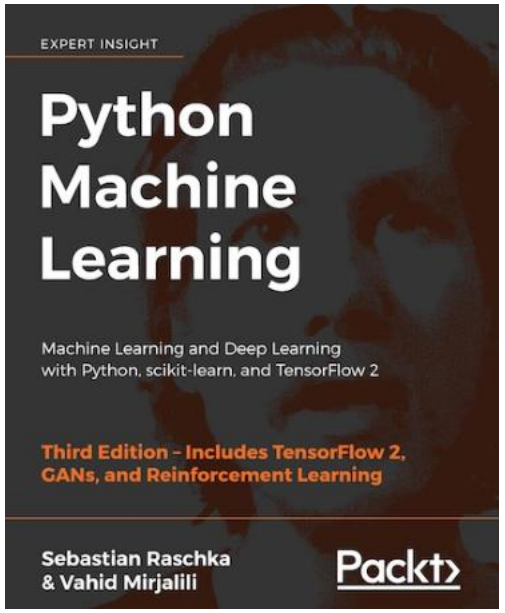
```
from autosklearn.classification import AutoSklearnClassifier

automl = AutoSklearnClassifier(time_left_for_this_task=120)
automl.fit(X_train, y_train)
```

Additional Resources

- **Books:**
  - "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron.
  - "Python Machine Learning" by Sebastian Raschka.
  - "The Hundred Page Machine Learning" by Andriy Burkov
- **Documentation and Tutorials:**
  - Official Scikit-Learn Documentation.







Thank You