5 Python Libraries for Auto Data Science

By Travis Tang
https://www.linkedin.com/in/travistang

sample code included!

Auto Data Exploration
Pandas Profiling

Auto Machine Learning
TPOT

Auto Deep Learning
AutoKeras

Auto Label Cleaning
Cleanlab

Automated Solution to Data Imbalance
Imbalanced-Learn

By Travis Tang
https://www.linkedin.com/in/travistang

Top 5 Automation Libraries for Machine Learning

By Travis Tang

1. Pandas Profiling: Automated data exploration
2. TPOT: AutoML
3. AutoKeras: Auto deep learning
4. Cleanlab: Auto label cleaning
5. imblearn: Auto data resampling

Get the code in this notebook

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

1. Automated Data Exploration: Pandas Profiling

```
import numpy as np
import pandas as pd
from pandas_profiling import ProfileReport
```

```
netflix_df = pd.read_csv('/kaggle/input/netflix-shows/netflix_titles.csv')
profile = ProfileReport(netflix_df, title="Pandas Profiling Report")
```

```
profile.to_notebook_iframe()
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]
Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]
Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]
```

Overview

2. AutoML with TPOT

```
from sklearn.preprocessing import LabelEncoder
iris_df = pd.read_csv('/kaggle/input/iris/Iris.csv')
iris_df = iris_df.set_index('Id')

encoder = LabelEncoder()
iris_df['Species'] = encoder.fit_transform(iris_df['Species'])
iris_df
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
Id					
1	5.1	3.5	1.4	0.2	0
2	4.9	3.0	1.4	0.2	0
3	4.7	3.2	1.3	0.2	0
4	4.6	3.1	1.5	0.2	0
5	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
146	6.7	3.0	5.2	2.3	2
147	6.3	2.5	5.0	1.9	2
148	6.5	3.0	5.2	2.0	2
149	6.2	3.4	5.4	2.3	2
150	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

```
from tpot import TPOTClassifier
from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test = train_test_split(iris_df.drop(columns=['Species']),
                                                    iris_df['Species'],
                                                    train_size=0.75, test_size=0.25)

pipeline_optimizer = TPOTClassifier(generations=5, population_size=20, cv=5,
                                    random_state=42, verbosity=2)
pipeline_optimizer.fit(X_train, y_train)
print(pipeline_optimizer.score(X_test, y_test))
pipeline_optimizer.export('tpot_exported_pipeline.py')
```

```
Optimization Progress:   0%|          | 0/120 [00:00<?, ?pipeline/s]
```

3. Auto deep learning hyperparameter tuning: AutoKeras

Code from https://autokeras.com/tutorial/image_classification/

```
Generation 4 - Current best internal CV score: 0.9644268774703558
```

```
!pip3 install autokeras
```

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import mnist

import autokeras as ak

# Load images
(x_train, y_train), (x_test, y_test) = mnist.load_data()
print(x_train.shape)  # (60000, 28, 28)
print(y_train.shape)  # (60000,)
print(y_train[:3])  # array([7, 2, 1], dtype=uint8)
```

```
2023-01-20 05:00:05.720207: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized witl
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-01-20 05:00:05.911065: W tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could not load c
2023-01-20 05:00:05.911118: I tensorflow/compiler/xla/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerro
2023-01-20 05:00:07.466988: W tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could not load c
2023-01-20 05:00:07.467232: W tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could not load c
2023-01-20 05:00:07.467248: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Cannot dlopen some Te
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 1s 0us/step
(60000, 28, 28)
(60000,)
[5 0 4]
```

```
# Initialize the image classifier.
clf = ak.ImageClassifier(overwrite=True, max_trials=1)
# Feed the image classifier with training data.
clf.fit(x_train, y_train, epochs=10)

# Predict with the best model.
predicted_y = clf.predict(x_test)
print(predicted_y)


# Evaluate the best model with testing data.
print(clf.evaluate(x_test, y_test))
```

```
Trial 1 Complete [00h 12m 42s]
val_loss: 0.03904299810528755

Best val_loss So Far: 0.03904299810528755
Total elapsed time: 00h 12m 42s
Epoch 1/10
1875/1875 [==============================] - 94s 50ms/step - loss: 0.1623 - accuracy: 0.9494
Epoch 2/10
1875/1875 [==============================] - 89s 47ms/step - loss: 0.0749 - accuracy: 0.9768
Epoch 3/10
1875/1875 [==============================] - 89s 47ms/step - loss: 0.0600 - accuracy: 0.9818
Epoch 4/10
1875/1875 [==============================] - 90s 48ms/step - loss: 0.0515 - accuracy: 0.9833
Epoch 5/10
1875/1875 [==============================] - 93s 49ms/step - loss: 0.0438 - accuracy: 0.9861
Epoch 6/10
1875/1875 [==============================] - 89s 48ms/step - loss: 0.0414 - accuracy: 0.9864
Epoch 7/10
1875/1875 [==============================] - 90s 48ms/step - loss: 0.0369 - accuracy: 0.9878
Epoch 8/10
1875/1875 [==============================] - 89s 47ms/step - loss: 0.0338 - accuracy: 0.9895
Epoch 9/10
1875/1875 [==============================] - 89s 48ms/step - loss: 0.0318 - accuracy: 0.9900
Epoch 10/10
1875/1875 [==============================] - 89s 48ms/step - loss: 0.0315 - accuracy: 0.9898
313/313 [==============================] - 4s 12ms/step
313/313 [==============================] - 4s 11ms/step
[['7']
 ['2']
 ['1']
 ...
 ['4']
 ['5']
 ['6']]
```

```
313/313 [==============================] - 4s 12ms/step - loss: 0.0326 - accuracy: 0.9898
[0.03263518586754799, 0.989799976348877]
```

4. Auto Detection of Dirty Labels: Cleanlab

```
!pip install -U skorch
```

```
Collecting skorch
  Downloading skorch-0.12.1-py3-none-any.whl (193 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 193.7/193.7 kB 4.8 MB/s eta 0:00:00a 0:00:01
Requirement already satisfied: scipy>=1.1.0 in /opt/conda/lib/python3.7/site-packages (from skorch) (1.7.3)
Requirement already satisfied: tabulate>=0.7.7 in /opt/conda/lib/python3.7/site-packages (from skorch) (0.9.0)
Requirement already satisfied: tqdm>=4.14.0 in /opt/conda/lib/python3.7/site-packages (from skorch) (4.64.0)
Requirement already satisfied: scikit-learn>=0.22.0 in /opt/conda/lib/python3.7/site-packages (from skorch) (1.0.2)
Requirement already satisfied: numpy>=1.13.3 in /opt/conda/lib/python3.7/site-packages (from skorch) (1.21.6)
Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/python3.7/site-packages (from scikit-learn>=0.22.0->skorch)
Requirement already satisfied: threadpoolctl>=2.0.0 in /opt/conda/lib/python3.7/site-packages (from scikit-learn>=0.22.0-
Installing collected packages: skorch
Successfully installed skorch-0.12.1
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system packag
```

```
!pip install cleanlab
```

```
Collecting cleanlab
  Downloading cleanlab-2.2.0-py3-none-any.whl (157 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 157.5/157.5 kB 4.0 MB/s eta 0:00:0000:01
Requirement already satisfied: scikit-learn>=0.18 in /opt/conda/lib/python3.7/site-packages (from cleanlab) (1.0.2)
Requirement already satisfied: numpy>=1.11.3 in /opt/conda/lib/python3.7/site-packages (from cleanlab) (1.21.6)
Requirement already satisfied: tqdm>=4.53.0 in /opt/conda/lib/python3.7/site-packages (from cleanlab) (4.64.0)
Requirement already satisfied: termcolor>=1.1.0 in /opt/conda/lib/python3.7/site-packages (from cleanlab) (1.1.0)
Requirement already satisfied: pandas>=1.0.0 in /opt/conda/lib/python3.7/site-packages (from cleanlab) (1.3.5)
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/lib/python3.7/site-packages (from pandas>=1.0.0->clea
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.7/site-packages (from pandas>=1.0.0->cleanlab) (202
Requirement already satisfied: scipy>=1.1.0 in /opt/conda/lib/python3.7/site-packages (from scikit-learn>=0.18->cleanlab)
Requirement already satisfied: threadpoolctl>=2.0.0 in /opt/conda/lib/python3.7/site-packages (from scikit-learn>=0.18->c
Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/python3.7/site-packages (from scikit-learn>=0.18->cleanlab)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-packages (from python-dateutil>=2.7.3->pandas>=1
Installing collected packages: cleanlab
Successfully installed cleanlab-2.2.0
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system packag
```

```
import torch
from torch import nn
from sklearn.datasets import fetch_openml
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import accuracy_score
from skorch import NeuralNetClassifier
```

```
# Import data
mnist = fetch_openml("mnist_784")  # Fetch the MNIST dataset

X = mnist.data.astype("float32").to_numpy() # 2D array (images are flattened into 1D)
X /= 255.0  # Scale the features to the [0, 1] range
X = X.reshape(len(X), 1, 28, 28)  # reshape into [N, C, H, W] for PyTorch

labels = mnist.target.astype("int64").to_numpy()  # 1D array of given labels
```

```
# Define neural network
class ClassifierModule(nn.Module):
    def __init__(self):
        super().__init__()

        self.cnn = nn.Sequential(
            nn.Conv2d(1, 6, 3),
            nn.ReLU(),
            nn.BatchNorm2d(6),
            nn.MaxPool2d(kernel_size=2, stride=2),
            nn.Conv2d(6, 16, 3),
            nn.ReLU(),
            nn.BatchNorm2d(16),
            nn.MaxPool2d(kernel_size=2, stride=2),
        )
        self.out = nn.Sequential(
            nn.Flatten(),
            nn.LazyLinear(128),
            nn.ReLU(),
```

```
            nn.Linear(128, 10),
            nn.Softmax(dim=-1),
        )

    def forward(self, X):
        X = self.cnn(X)
        X = self.out(X)
        return X
```

```
# Train the model
model_skorch = NeuralNetClassifier(ClassifierModule)
```

```
# Compute out-of-sample predicted probabilities
num_crossval_folds = 3  # for efficiency; values like 5 or 10 will generally work better
pred_probs = cross_val_predict(
    model_skorch,
    X,
    labels,
    cv=num_crossval_folds,
    method="predict_proba",
)
```

```
/opt/conda/lib/python3.7/site-packages/torch/nn/modules/lazy.py:178: UserWarning: Lazy modules are a new feature under he
  warnings.warn('Lazy modules are a new feature under heavy development '
  epoch    train_loss    valid_acc    valid_loss     dur
-------  ------------  -----------  ------------  ------
      1        0.7401       0.9085        0.3317  4.0437
      2        0.2202       0.9432        0.2017  4.4079
      3        0.1525       0.9545        0.1548  4.3429
      4        0.1220       0.9604        0.1308  4.0353
      5        0.1040       0.9646        0.1161  3.9484
      6        0.0921       0.9672        0.1064  4.0506
      7        0.0833       0.9691        0.0990  3.9404
      8        0.0764       0.9708        0.0932  3.8551
      9        0.0709       0.9726        0.0886  3.9266
     10        0.0663       0.9732        0.0851  3.9215
/opt/conda/lib/python3.7/site-packages/torch/nn/modules/lazy.py:178: UserWarning: Lazy modules are a new feature under he
  warnings.warn('Lazy modules are a new feature under heavy development '
  epoch    train_loss    valid_acc    valid_loss     dur
-------  ------------  -----------  ------------  ------
      1        0.7544       0.9239        0.2825  4.0390
      2        0.1990       0.9479        0.1845  4.2142
      3        0.1394       0.9578        0.1468  3.9077
      4        0.1123       0.9627        0.1264  4.1081
      5        0.0963       0.9678        0.1129  4.1166
      6        0.0853       0.9703        0.1036  4.0916
      7        0.0772       0.9721        0.0967  3.9913
      8        0.0707       0.9726        0.0914  4.1284
      9        0.0654       0.9745        0.0871  4.1670
     10        0.0609       0.9751        0.0835  4.0092
/opt/conda/lib/python3.7/site-packages/torch/nn/modules/lazy.py:178: UserWarning: Lazy modules are a new feature under he
  warnings.warn('Lazy modules are a new feature under heavy development '
  epoch    train_loss    valid_acc    valid_loss     dur
-------  ------------  -----------  ------------  ------
      1        0.7643       0.9308        0.2861  3.9461
      2        0.2134       0.9506        0.1873  3.8716
      3        0.1549       0.9585        0.1509  3.8670
      4        0.1262       0.9607        0.1352  3.9094
      5        0.1082       0.9640        0.1212  3.8866
      6        0.0954       0.9668        0.1119  3.8642
      7        0.0856       0.9684        0.1047  4.1236
      8        0.0778       0.9697        0.0991  3.9201
      9        0.0714       0.9714        0.0942  3.8973
     10        0.0662       0.9724        0.0900  3.9248
```

```
predicted_labels = pred_probs.argmax(axis=1)
acc = accuracy_score(labels, predicted_labels)
print(f"Cross-validated estimate of accuracy on held-out data: {acc}")
```

```
Cross-validated estimate of accuracy on held-out data: 0.9765142857142857
```

```
from cleanlab.filter import find_label_issues

ranked_label_issues = find_label_issues(
    labels,
    pred_probs,
    return_indices_ranked_by="self_confidence",
)

print(f"Cleanlab found {len(ranked_label_issues)} label issues.")
print(f"Top 15 most likely label errors: \n {ranked_label_issues[:15]}")
```

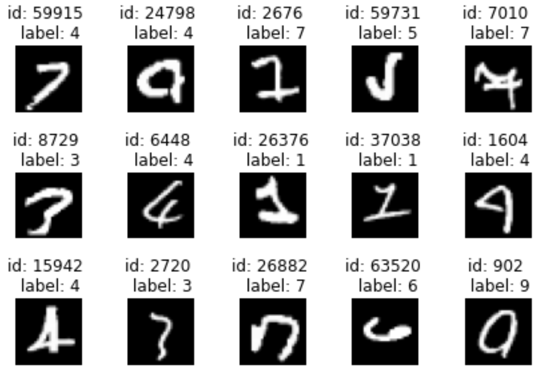
```
Cleanlab found 125 label issues.
Top 15 most likely label errors:
 [59915 24798  2676 59731  7010  8729  6448 26376 37038  1604 15942  2720
 26882 63520   902]
```

```
import matplotlib.pyplot as plt

def plot_examples(id_iter, nrows=1, ncols=1):
    for count, id in enumerate(id_iter):
        plt.subplot(nrows, ncols, count + 1)
        plt.imshow(X[id].reshape(28, 28), cmap="gray")
        plt.title(f"id: {id} \n label: {labels[id]}")
        plt.axis("off")

    plt.tight_layout(h_pad=2.0)
```

```
# Plot data that are incorrectly labeled
plot_examples(ranked_label_issues[range(15)], 3, 5)
```



5. Auto Balancing of Imbalanced Data with Imbalanced-Learn

```
from collections import Counter

from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

from imblearn.datasets import make_imbalance
from imblearn.metrics import classification_report_imbalanced
from imblearn.pipeline import make_pipeline
from imblearn.under_sampling import NearMiss

print(__doc__)

RANDOM_STATE = 42

# Create a folder to fetch the dataset
iris = load_iris()
X, y = make_imbalance(
    iris.data,
    iris.target,
    sampling_strategy={0: 25, 1: 50, 2: 50},
    random_state=RANDOM_STATE,
)

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

print(f"Training target statistics: {Counter(y_train)}")
print(f"Testing target statistics: {Counter(y_test)}")

# Create a pipeline
pipeline = make_pipeline(
    NearMiss(version=2), StandardScaler(), LogisticRegression(random_state=RANDOM_STATE)
)
pipeline.fit(X_train, y_train)

# Classify and report the results
print(classification_report_imbalanced(y_test, pipeline.predict(X_test)))
```

```
Automatically created module for IPython interactive environment
Training target statistics: Counter({1: 38, 2: 38, 0: 17})
Testing target statistics: Counter({1: 12, 2: 12, 0: 8})
                   pre       rec       spe        f1       geo       iba       sup

          0       1.00      1.00      1.00      1.00      1.00      1.00         8
          1       0.88      0.58      0.95      0.70      0.74      0.53        12
          2       0.69      0.92      0.75      0.79      0.83      0.70        12

avg / total       0.84      0.81      0.89      0.81      0.84      0.71        32
```