Week 3 - Ungraded Lab: Data Labeling

Welcome to the ungraded lab for week 3 of Machine Learning Engineering for Production. In this lab, you will see how the data labeling process affects the performance of a classification model. Labeling data is usually a very labor intensive and costly task but it is of great importance.

As you saw in the lectures there are many ways to label data, this is dependant on the strategy used. Recall the example with the iguanas, all of the following are valid labeling alternatives but they clearly follow different criteria.

You can think of every labeling strategy as a result of different labelers following different labeling rules. If your data is labeled by people using different criteria this will have a negative impact on your learning algorithm. It is desired to have consistent labeling across your dataset.

This lab will touch on the effect of labeling strategies from a slighlty different angle. You will explore how different strategies affect the performance of a machine learning model by simulating the process of having different labelers label the data. This, by defining a set of rules and performing automatic labeling based on those rules.

The main objective of this ungraded lab is to compare performance across labeling options to understand the role that good labeling plays on the performance of Machine Learning models, these options are:

1. Randomly generated labels (performance lower bound)
2. Automatic generated labels based on three different label strategies
3. True labels (performance upper bound)

Although the example with the iguanas is a computer vision task, the same concepts regarding labeling can be applied to other types of data. In this lab you will be working with text data, concretely you will be using a dataset containing comments from the 2015 top 5 most popular Youtube videos. Each comment has been labeled as `spam` or `not_spam` depending on its contents.

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Loading the dataset

The dataset consists of 5 CSV files, one for each video. Pandas `DataFrame` are very powerful to handle data in CSV format. The following helper function will load the data using pandas:

```
def load_labeled_spam_dataset():
    """Load labeled spam dataset."""

    # Path where csv files are located
    base_path = "./data/"

    # List of csv files with full path
    csv_files = [os.path.join(base_path, csv) for csv in os.listdir(base_path)]

    # List of dataframes for each file
    dfs = [pd.read_csv(filename) for filename in csv_files]

    # Concatenate dataframes into a single one
    df = pd.concat(dfs)

    # Rename columns
    df = df.rename(columns={"CONTENT": "text", "CLASS": "label"})

    # Set a seed for the order of rows
    df = df.sample(frac=1, random_state=824)

    return df.reset_index()
```

```
# Save the dataframe into the df_labeled variable
df_labeled = load_labeled_spam_dataset()
```

To have a feeling of how the data is organized, let's inspect the top 5 rows of the data:

```
# Take a look at the first 5 rows
df_labeled.head()
```

	index	COMMENT_ID	AUTHOR	DATE	text	label
0	86	z12uzb3oxy23cbikz23qsjczaxz5wh1t5	luisel Tutoriales Gameplays (luisel Tutoriales)	2015-05-23T18:56:07.484000	music yeah	0
1	291	z122z5pa2wyofbjj304cgfwrrmvjgn0pohc	Mia Aspinall	2014-11-08T10:30:35	2 billion views, only 2 million shares	0
2	142	z13bttm5gxecfvrqq04cevkpuxj5s1u5ys40k	Vane Cavazos	2014-09-20T15:39:44	Katycat! https://m.facebook.com /profile.php?id...	1
3	147	z132zl1rupqcylbep23jgfig3um3ct5vv	KatyPerry TheQueenOfPop	2014-09-22T06:50:24	-->ATTENTION KATYCATS! Katy leads with 7 no...	1

Further inspection and preprocessing

Checking for data imbalance

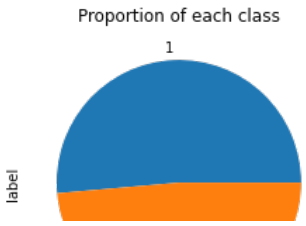
It is fairly common to assume that the data you are working on is balanced. This means that the dataset contains a similar proportion of examples for all classes. Before moving forward let's actually test this assumption:

```
# Print actual value count
print(f"Value counts for each class:\n\n{df_labeled.label.value_counts()}\n")

# Display pie chart to visually check the proportion
df_labeled.label.value_counts().plot.pie(y='label', title='Proportion of each class')
plt.show()
```

```
Value counts for each class:

1    1005
0     951
Name: label, dtype: int64
```



```
# Drop unused columns
df_labeled = df_labeled.drop(['index', 'COMMENT_ID', 'AUTHOR', 'DATE'], axis=1)

# Look at the cleaned dataset
df_labeled.head()
```

	text	label
0	music yeah	0
1	2 billion views, only 2 million shares	0
2	Katycat! https://m.facebook.com/profile.php?id...	1
3	-->ATTENTION KATYCATS! Katy leads with 7 no...	1
4	I wanted to know the name of the guy that danc...	0

Now the dataset only includes the information you are going to use moving forward.

Splitting the dataset

Before jumping to the data labeling section let's split the data into training and test sets so you can use the latter to measure the performance of models that were trained using data labeled through different methods. As a safety measure when doing this split, remember to use stratification so the proportion of classes is maintained within each split.

```
from sklearn.model_selection import train_test_split

# Save the text into the X variable
X = df_labeled.drop("label", axis=1)

# Save the true labels into the y variable
y = df_labeled["label"]

# Use 1/5 of the data for testing later
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Print number of comments for each set
print(f"There are {X_train.shape[0]} comments for training.")
print(f"There are {X_test.shape[0]} comments for testing")
```

Both, the training and test sets a balanced proportion of examples per class. So, the code successfully implemented stratification.

Let's get going!

Data Labeling

Establishing performance lower and upper bounds for reference

To properly compare different labeling strategies you need to establish a baseline for model accuracy, in this case you will establish both a lower and an upper bound to compare against.

Calculate accuracy of a labeling strategy

CountVectorizer is a handy tool included in the sklearn ecosystem to encode text based data.

For more information on how to work with text data using sklearn check out this resource.

[ ] ↳ *5 cells hidden*

Random Labeling

Generating random labels is a natural way to establish a lower bound. You will expect that any successful alternative labeling model to outperform randomly generated labels.

First iteration - Define some rules

For this first iteration you will create three rules based on the intuition of common patterns that appear on spam comments. The rules are simple, classify as SPAM if any of the following patterns is present within the comment or NO_LABEL otherwise:

- `free` - spam comments usually lure users by promoting free stuff
- `subs` - spam comments tend to ask users to subscribe to some website or channel
- `http` - spam comments include links very frequently

[ ] ↳ *12 cells hidden*