Introducing the challenge

CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON

Peter Bull
Co-founder of DrivenData

Introducing the challenge

- Learn from the expert who won DrivenData's challenge
  - Natural language processing
  - Feature engineering
  - Efficiency boosting hashing tricks
- Use data to have a social impact

Introducing the challenge

- Budgets for schools are huge, complex, and not standardized
  - Hundreds of hours each year are spent manually labelling
- Goal: Build a machine learning algorithm that can automate the process
- Budget data
  - Line-item: "Algebra books for 8th grade students"
  - Labels: "Textbooks", "Math", "Middle School"
- This is a supervised learning problem

Over 100 target variables!

- This is a classification problem

- Pre_K:
  - NO_LABEL
  - Non PreK
  - PreK
- Reporting:
  - NO_LABEL
  - Non-School
  - School
- Sharing:
  - Leadership & Management
  - NO_LABEL
  - School Reported
- Student_Type:
  - Alternative
  - At Risk
  - ...

Over 100 target variables!

- This is a classification problem
  - Pre_K:
    - NO_LABEL
    - Non PreK
    - PreK
  - Reporting:
    - NO_LABEL
    - Non-School
    - School
  - Sharing:
    - Leadership & Management
    - NO_LABEL
    - School Reported
  - Student_Type:
    - Alternative
    - At Risk
    - …

Over 100 target variables!

- This is a classification problem
    - Pre_K:
        - NO_LABEL
        - Non PreK
        - PreK
    - Reporting:
        - NO_LABEL
        - Non-School
        - School
    - Sharing:
        - Leadership & Management
        - NO_LABEL
        - School Reported
    - Student_Type:
        - Alternative
        - At Risk
        - ...

How we can help

	Function__Aides Compensation	Function__Career & Academic Counseling	Function__Communications	...	Use__O&M	Use__Pupil Services & Enrichment	Use__Untracked Budget Set-Aside
180042	0.027027	0.027027	0.027027	...	0.125	0.125	0.125
28872	0.027027	0.027027	0.027027	...	0.125	0.125	0.125
186915	0.027027	0.027027	0.027027	...	0.125	0.125	0.125
412396	0.027027	0.027027	0.027027	...	0.125	0.125	0.125
427740	0.027027	0.027027	0.027027	...	0.125	0.125	0.125

How we can help

	Function__Aides Compensation	Function__Career & Academic Counseling	Function__Communications	...	Use__O&M	Use__Pupil Services & Enrichment	Use__Untracked Budget Set-Aside
180042	0.027027	0.027027	0.027027	...	0.125	0.125	0.125
28872	0.027027	0.027027	0.027027	...	0.125	0.125	0.125
186915	0.027027	0.027027	0.027027	...	0.125	0.125	0.125
412396	0.027027	0.027027	0.027027	...	0.125	0.125	0.125
427740	0.027027	0.027027	0.027027	...	0.125	0.125	0.125

- Predictions will be **probabilities** for each label

Let's practice!

CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON

Exploring the data

CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON

Peter Bull
Co-founder of DrivenData

A column for each possible value

	Eyes	Hair
Jamal	Brown	Curly
Luisa	Brown	Straight
Jenny	Blue	Wavy
Max	Blue	Straight

A column for each possible value

	Eyes	Hair
Jamal	Brown	Curly
Luisa	Brown	Straight
Jenny	Blue	Wavy
Max	Blue	Straight

	Eyes_Blue	Eyes_Brown	Hair_Curly	Hair_Straight	Hair_Wavy
Jamal	0	1	1	0	0
Luisa	0	1	0	1	0
Jenny	1	0	0	0	1
Max	1	0	0	1	0

Load and preview the data

```
import pandas as pd
sample_df = pd.read_csv('sample_data.csv')
sample_df.head()
```

```
  label    numeric     text  with_missing
0     a  -4.167578      bar     -4.084883
1     a  -0.562668                2.043464
2     a -21.361961              -33.315334
3     b  16.402708  foo bar     30.884604
4     a -17.934356      foo    -27.488405
```

Summarize the data

```
sample_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100 entries, 0 to 99
Data columns (total 4 columns):
label           100 non-null object
numeric         100 non-null float64
text            100 non-null object
with_missing    95  non-null float64
dtypes: float64(2), object(2)
memory usage: 3.9+ KB
```

Summarize the data

```
sample_df.describe()
```

```
          numeric  with_missing
count  100.000000     95.000000
mean    -1.037411      1.275189
std     10.422602     17.386723
min    -26.594495    -42.210641
25%     -6.952244     -8.312870
50%     -0.653688      1.733997
75%      5.398819     11.777888
max     22.922080     41.967536
```

Let's practice!

CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON

Looking at the datatypes

CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON

Peter Bull
Co-founder of DrivenData

Objects instead of categories

```
sample_df['label'].head()
```

```
0     a
1     a
2     a
3     b
4     a
Name: label, dtype: object
```

Encode labels as categories

- ML algorithms work on numbers, not strings
  - Need a numeric representation of these strings
- Strings can be slow compared to numbers
- In pandas, `category` dtype encodes categorical data numerically
  - Can speed up code

Encode labels as categories (sample data)

```
sample_df.label.head(2)
```

```
0     a
1     b
Name: label, dtype: object
```

```
sample_df.label = sample_df.label.astype('category')
sample_df.label.head(2)
```

```
0     a
1     b
Name: label, dtype: category
Categories (2, object): [a, b]
```

Dummy variable encoding

```
dummies = pd.get_dummies(sample_df[['label']], prefix_sep='_')
dummies.head(2)
```

```
   label_a  label_b
0        1        0
1        0        1
```

- Also called a `binary indicator` representation

Lambda functions

- Alternative to `def` syntax
- Easy way to make simple, one-line functions

```
square = lambda x: x*x
square(2)
```

```
4
```

Encode labels as categories

- In the sample dataframe, we only have one relevant column
- In the budget data, there are multiple columns that need to be made categorical

Encode labels as categories

```
categorize_label = lambda x: x.astype('category')
sample_df.label = sample_df[['label']].apply(categorize_label, axis=0)
sample_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100 entries, 0 to 99
Data columns (total 4 columns):
label           100 non-null category
numeric         100 non-null float64
text            100 non-null object
with_missing    95 non-null float64
dtypes: category(1), float64(2), object(1)
memory usage: 3.2+ KB
```

Let's practice!

CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON

How do we measure success?

CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON

Peter Bull
Co-founder of DrivenData

How do we measure success?

- Accuracy can be misleading when classes are imbalanced
  - Legitimate email: 99%, Spam: 1%
  - Model that never predicts spam will be 99% accurate!
- Metric used in this problem: log loss
  - It is a loss function
  - Measure of error
  - Want to minimize the error (unlike accuracy)

Log loss binary classification

- Log loss for **binary** classification

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1 - y_i)\log(1 - p_i))$$

Log loss binary classification

- Log loss for **binary** classification

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1 - y_i)\log(1 - p_i))$$

- Actual value: y = {1=yes, 0=no}

Log loss binary classification

- Log loss for **binary** classification

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1 - y_i)\log(1 - p_i))$$

- Actual value: y = {1=yes, 0=no}
- Prediction (probability that the value is 1): p

Log loss binary classification

- Log loss for binary classification

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1 - y_i)\log(1 - p_i))$$

- Actual value: y = {1=yes, 0=no}
- Prediction (probability that the value is 1): p

Log loss binary classification

- Log loss for binary classification

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1 - y_i)\log(1 - p_i))$$

- Actual value: y = {1=yes, 0=no}
- Prediction (probability that the value is 1): p

Log loss binary classification: example

$$logloss_{(\mathrm{N=1})} = y\log(p) + (1-y)\log(1-p)$$

- True label = 0
- Model confidently predicts 1 (with p = 0.90)

$$Logloss = (1-y)\ log(1-p)$$

$$= log(1-0.9)$$

$$= log(0.1)$$

$$= 2.30$$

Log loss binary classification: example

$$logloss_{(\mathrm{N=1})} = y\log(p) + (1-y)\log(1-p)$$

- True label = 1
- Model predicts 0 (with p = 0.50)
- Log loss = 0.69
- Better to be less confident than confident and wrong

Computing log loss with NumPy

- logloss.py

```
import numpy as np
def compute_log_loss(predicted, actual, eps=1e-14):
    """ Computes the logarithmic loss between predicted and
        actual when these are 1D arrays.

        :param predicted: The predicted probabilities as floats between 0-1
        :param actual: The actual binary labels. Either 0 or 1.
        :param eps (optional): log(0) is inf, so we need to offset our
                               predicted values slightly by eps from 0 or 1.
    """
    predicted = np.clip(predicted, eps, 1 - eps)
    loss = -1 * np.mean(actual * np.log(predicted)
              + (1 - actual)
              * np.log(1 - predicted))

    return loss
```

Computing log loss with NumPy

```
compute_log_loss(predicted=0.9, actual=0)
```

```
2.3025850929940459
```

```
compute_log_loss(predicted=0.5, actual=1)
```

```
0.69314718055994529
```

Let's practice!

CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON