Learning from the expert: processing

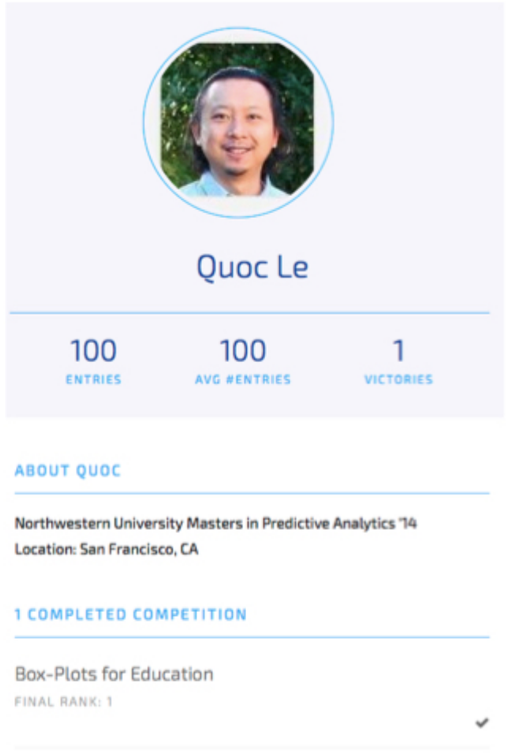
CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON

Peter Bull
Co-founder of DrivenData

DataCamp

Learning from the expert

- Text processing
- Statistical methods
- Computational efficiency



Learning from the expert: text preprocessing

- NLP tricks for text data
  - Tokenize on punctuation to avoid hyphens, underscores, etc.
  - Include unigrams **and** bi-grams in the model to capture important information involving multiple tokens - e.g., "middle school"

N-grams and tokenization

```
vec = CountVectorizer(token_pattern=TOKENS_ALPHANUMERIC,
                      ngram_range=(1, 2))
```

- Simple changes to `CountVectorizer`
- alphanumeric tokenization
- `ngram_range=(1, 2)`

Range of n-grams in scikit-learn

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

```
Pipeline(steps=[('union', FeatureUnion(n_jobs=1,
       transformer_list=[('numeric_features',
Pipeline(steps=[('selector',
FunctionTransformer(accept_sparse=False,
          func=<function <lambda> at 0x11441f7b8>, pass_y=False,
          validate=False)), ('imputer', Imputer(axis=0, copy=True,
missing_valu...=None,
          solver='liblinear',
          tol=0.0001,
          verbose=0, warm_start=False),
          n_jobs=1))])
```

Range of n-grams in scikit-learn

```
holdout = pd.read_csv('HoldoutData.csv', index_col=0)

predictions = pl.predict_proba(holdout)

prediction_df = pd.DataFrame(columns=pd.get_dummies(
                df[LABELS]).columns, index=holdout.index,
                data=predictions)

prediction_df.to_csv('predictions.csv')

score = score_submission(pred_path='predictions.csv')
```

Let's practice!

CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON

DataCamp

Learning from the expert: a stats trick

CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON

Peter Bull
Co-founder of DrivenData

DataCamp

Learning from the expert: interaction terms

- Statistical tool that the winner used: interaction terms
- Example
  - English teacher for 2nd grade
  - 2nd grade - budget for English teacher
- Interaction terms mathematically describe when tokens appear together

Interaction terms: the math

$$\beta_1 x_1 + \beta_2 x_2 + \beta_3 (x_1 \times x_2)$$

Interaction terms: the math

$$\beta_1 x_1 + \beta_2 x_2 + \beta_3 (x_1 \times x_2)$$

Interaction terms: the math

$$\beta_1 x_1 + \beta_2 x_2 + \beta_3 (x_1 \times x_2)$$

X1	X2
0	1
1	1

Interaction terms: the math

$$\beta_1 x_1 + \beta_2 x_2 + \beta_3 (x_1 \times x_2)$$

X1	X2	X3
0	1	X1*X2 = 0*1 = 0
1	1	X1*X2 = 1*1 = 1

Adding interaction features with scikit-learn

```
from sklearn.preprocessing import PolynomialFeatures
x
```

```
   x1  x2
a   0   1
b   1   1
```

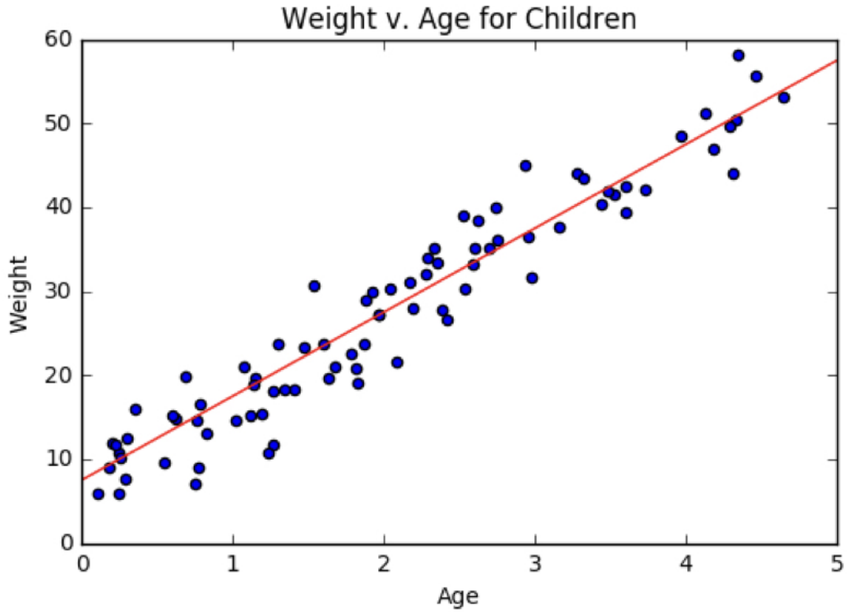
```
interaction = PolynomialFeatures(degree=2,
                                 interaction_only=True,
                                 include_bias=False)

interaction.fit_transform(x)
```

```
array([[ 0.,  1.,  0.],
       [ 1.,  1.,  1.]])
```

A note about bias terms

- Bias term allows model to have non-zero y value when x value is zero



Sparse interaction features

```
SparseInteractions(degree=2).fit_transform(x).toarray()
```

```
array([[ 0.,  1.,  0.],
       [ 1.,  1.,  1.]])
```

- The number of interaction terms grows exponentially
- Our vectorizer saves memory by using a sparse matrix
- `PolynomialFeatures` does not support sparse matrices
- We have provided `SparseInteractions` to work for this problem

Let's practice!

CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON

DataCamp

Learning from the expert: the winning model

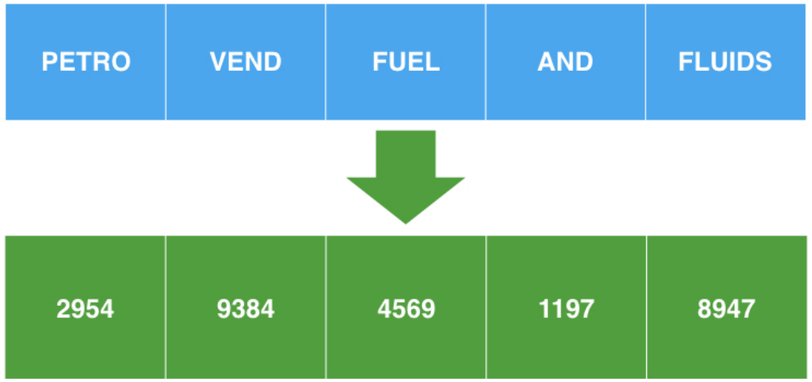
CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON

Peter Bull
Co-founder of DrivenData

DataCamp

Learning from the expert: hashing trick

- Adding new features may cause enormous increase in array size
- Hashing is a way of increasing memory efficiency



- Hash function limits possible outputs, fixing array size

When to use the hashing trick

- Want to make array of features as small as possible
  - Dimensionality reduction
  - Particularly useful on large datasets
  - e.g., lots of text data!

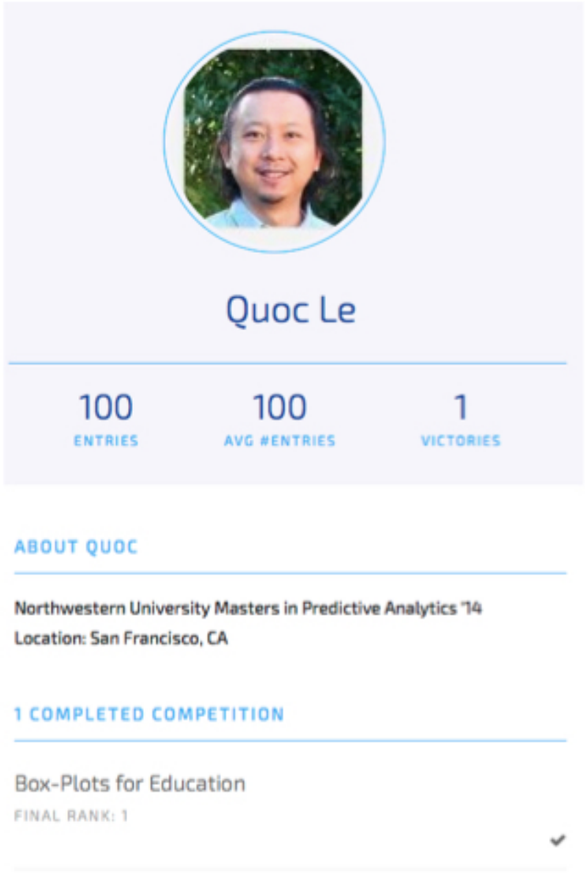
Implementing the hashing trick in scikit-learn

```
from sklearn.feature_extraction.text import HashingVectorizer

vec = HashingVectorizer(norm=None,
                        non_negative=True,
                        token_pattern=TOKENS_ALPHANUMERIC,
                        ngram_range=(1, 2))
```

The model that won it all

- You now know all the expert moves to make on this dataset
  - NLP: Range of n-grams, punctuation tokenization
  - Stats: Interaction terms
  - Computation: Hashing trick
- What class of model was used?



The model that won it all

- And the winning model was…
- Logistic regression!
  - Carefully create features
  - Easily implemented tricks
  - Favor simplicity over complexity and see how far it takes you!

Let's practice!

CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON

DataCamp

Next steps and the social impact of your work

CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON

Peter Bull
Co-founder of DrivenData

DataCamp

Can you do better?

- You've seen the flexibility of the pipeline steps
- Quickly test ways of improving your submission
  - NLP: Stemming, stop-word removal
  - Model: RandomForest, k-NN, Naïve Bayes
  - Numeric Preprocessing: Imputation strategies
  - Optimization: Grid search over pipeline objects
  - Experiment with new scikit-learn techniques
- Work with the full dataset at DrivenData!

Hundreds of hours saved

- Make schools more efficient by improving their budgeting decisions
- Saves hundreds of hours each year that humans spent labeling line items
- Can spend more time on the decisions that really matter

DrivenData: Data Science to save the world

- Other ways to use data science to have a social impact at www.drivendata.org
  - Improve your data science skills while helping meaningful organizations thrive
  - Win some cash prizes while you're at it!

Let's practice!

CASE STUDY: SCHOOL BUDGETING WITH MACHINE LEARNING IN PYTHON

DataCamp