11/2017
Datacamp - Machine Learning with the Experts: School Budgets (Data Scientist Track with Python)
[Machine Learning with the Experts: School Budgets]

Course Description

Data science isn't just for predicting ad-clicks-it's also useful for social impact! This course is a case study from a machine learning competition on DrivenData. You'll explore a problem related to school district budgeting. By building a model to automatically classify items in a school's budget, it makes it easier and faster for schools to compare their spending with other schools. In this course, you'll begin by building a baseline model that is a simple, first-pass approach. In particular, you'll do some natural language processing to prepare the budgets for modeling. Next, you'll have the opportunity to try your own techniques and see how they compare to participants from the competition. Finally, you'll see how the winner was able to combine a number of expert techniques to build the most accurate model.

Part 1 : Exploring the raw data

In this chapter, you'll be introduced to the problem you'll be solving in this course. How do you accurately classify line-items in a school budget based on what that money is being used for? You will explore the raw text and numeric values in the dataset, both quantitatively and visually. And you'll learn how to measure success when trying to predict class labels for each row of the dataset.

What category of problem is this?

You're no novice to data science, but let's make sure we agree on the basics.

As Peter from DrivenData explained in the video, you're going to be working with school district budget data. This data can be classified in many ways according to certain labels, e.g. Function: Career & Academic Counseling, or Position_Type: Librarian.

Your goal is to develop a model that predicts the probability for each possible label by relying on some correctly labeled examples.

What type of machine learning problem is this?

Possible Answers : => 4

- Reinforcement Learning, because the model is learning from the data through a system of rewards and punishments.
- Unsupervised Learning, because the model doesn't output labels with certainty.
- Unsupervised Learning, because not all data is correctly classified to begin with.
- Supervised Learning, because the model will be trained using labeled examples.

```
```

Results :

Yes! Using correctly labeled budget line items to train means this is a supervised learning problem.

What is the goal of the algorithm?

As you know from previous courses, there are different types of supervised machine learning problems. In this exercise you will tell us what type of supervised machine learning problem this is, and why you think so.

Remember, your goal is to correctly label budget line items by training a supervised model to predict the probability of each possible label, taking most probable label as the correct label.

Possible Answers : => 2

- Regression, because the model will output probabilities.
- Classification, because predicted probabilities will be used to select a label class.
- egression, because probabilities take a continuous value between 0 and 1.
- Classification, because the model will output probabilities.

```
```

Results :

Nice! Specifically, we have ourselves a multi-class-multi-label classification problem (quite a mouthful!), because there are 9 broad categories that each take on many possible sub-label instances.

Loading the data

Now it's time to check out the dataset! You'll use pandas (which has been pre-imported as pd) to load your data into a DataFrame and then do some Exploratory Data Analysis (EDA) of it.

The training data is available as TrainingData.csv. Your first task is to load it into a DataFrame in the IPython Shell using pd.read_csv() along with the keyword argument index_col=0.

Use methods such as .info(), .head(), and .tail() to explore the budget data and the properties of the features and labels.

Some of the column names correspond to features - descriptions of the budget items - such as the Job_Title_Description column. The values in this column tell us if a budget item is for a teacher, custodian, or other employee.

Some columns correspond to the budget item labels you will be trying to predict with your model. For example, the Object_Type column describes whether the budget item is related classroom supplies, salary, travel expenses, etc.

Use df.info() in the IPython Shell to answer the following questions:

- How many rows are there in the training data?
- How many columns are there in the training data?
- How many non-null entries are in the Job_Title_Description column?

Possible Answers : => 3

- 25 rows, 1560 columns, 1560 non-null entries in Job_Title_Description.
- 225 rows, 1560 columns, 1131 non-null entries in Job_Title_Description.
- 21560 rows, 25 columns, 1131 non-null entries in Job_Title_Description.
- 21560 rows, 25 columns, 1560 non-null entries in Job_Title_Description.

Results :

```
In [4]: import pandas as pd

In [5]: df = pd.read_csv('TrainingData.csv',index_col=0)

In [6]: df.shape
Out[6]: (1560, 25)

In [7]: df.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1560 entries, 198 to 101861
Data columns (total 25 columns):
Function                  1560 non-null object
Use                       1560 non-null object
Sharing                   1560 non-null object
Reporting                 1560 non-null object
Student_Type              1560 non-null object
Position_Type             1560 non-null object
Object_Type               1560 non-null object
Pre_K                     1560 non-null object
Operating_Status          1560 non-null object
Object_Description        1461 non-null object
Text_2                    382 non-null object
SubFund_Description       1183 non-null object
Job_Title_Description     1131 non-null object
Text_3                    677 non-null object
Text_4                    193 non-null object
Sub_Object_Description    364 non-null object
Location_Description      874 non-null object
FTE                       449 non-null float64
Function_Description      1340 non-null object
Facility_or_Department    252 non-null object
Position_Extra            1026 non-null object
Total                     1542 non-null float64
Program_Description       1192 non-null object
Fund_Description          819 non-null object
Text_1                    1132 non-null object
dtypes: float64(2), object(23)
memory usage: 316.9+ KB

In [8]: df.describe()
Out[8]:
              FTE         Total
count  449.000000  1.542000e+03
mean     0.493532  1.446867e+04
std      0.452844  7.916752e+04
min     -0.002369 -1.044084e+06
25%           NaN           NaN
50%           NaN           NaN
75%           NaN           NaN
max      1.047222  1.367500e+06

In [9]: df.head()
Out[9]:
```

```
                Function          Use          Sharing   Reporting  \
198                 NO_LABEL     NO_LABEL         NO_LABEL    NO_LABEL
209   Student Transportation     NO_LABEL  Shared Services  Non-School
750     Teacher Compensation  Instruction  School Reported      School
931                 NO_LABEL     NO_LABEL         NO_LABEL    NO_LABEL
1524                NO_LABEL     NO_LABEL         NO_LABEL    NO_LABEL

     Student_Type Position_Type               Object_Type     Pre_K  \
198      NO_LABEL      NO_LABEL                  NO_LABEL  NO_LABEL
209      NO_LABEL      NO_LABEL    Other Non-Compensation  NO_LABEL
750   Unspecified       Teacher  Base Salary/Compensation  Non PreK
931      NO_LABEL      NO_LABEL                  NO_LABEL  NO_LABEL
1524     NO_LABEL      NO_LABEL                  NO_LABEL  NO_LABEL

       Operating_Status              Object_Description  \
198       Non-Operating                  Supplemental *
209   PreK-12 Operating  REPAIR AND MAINTENANCE SERVICES
750   PreK-12 Operating     Personal Services - Teachers
931       Non-Operating                General Supplies
1524      Non-Operating           Supplies and Materials

          ...                      Sub_Object_Description  \
198           ...           Non-Certificated Salaries And Wages
209           ...                                       NaN
750           ...                                       NaN
931           ...                          General Supplies
1524          ...                    Supplies And Materials

     Location_Description  FTE                  Function_Description  \
198                   NaN  NaN  Care and Upkeep of Building Services
209       ADMIN. SERVICES  NaN              STUDENT TRANSPORT SERVICE
750                   NaN  1.0                                    NaN
931                   NaN  NaN                            Instruction
1524                  NaN  NaN            Other Community Services *

      Facility_or_Department Position_Extra     Total  \
198                      NaN            NaN  -8291.86
209                      NaN            NaN    618.29
750                      NaN        TEACHER  49768.82
931   Instruction And Curriculum            NaN     -1.02
1524                     NaN            NaN   2304.43

                                Program_Description  \
198                                             NaN
209                           PUPIL TRANSPORTATION
750                          Instruction - Regular
931   "Title I, Part A Schoolwide Activities Related...
1524                                            NaN

                                  Fund_Description               Text_1
198   Title I - Disadvantaged Children/Targeted Assi...    TITLE I CARRYOVER
209                                       General Fund                  NaN
750                               General Purpose School                NaN
931                                General Operating Fund               NaN
1524  Title I - Disadvantaged Children/Targeted Assi...   TITLE I PI+HOMELESS

[5 rows x 25 columns]

In [10]: df.tail()
Out[10]:
```

```
                   Function          Use                 Sharing  \
344986   Substitute Compensation  Instruction          School Reported
384803                 NO_LABEL     NO_LABEL                NO_LABEL
224382   Substitute Compensation  Instruction          School Reported
305347  Facilities & Maintenance          O&M  Leadership & Management
101861      Teacher Compensation  Instruction          School Reported

     Reporting       Student_Type Position_Type  \
344986      School        Unspecified    Substitute
384803    NO_LABEL           NO_LABEL      NO_LABEL
224382      School  Special Education    Substitute
305347  Non-School             Gifted     Custodian
101861      School            Poverty       Teacher

                  Object_Type     Pre_K   Operating_Status  \
344986                   Benefits  NO_LABEL  PreK-12 Operating
384803                   NO_LABEL  NO_LABEL      Non-Operating
224382     Substitute Compensation  NO_LABEL  PreK-12 Operating
305347  Other Compensation/Stipend  Non PreK  PreK-12 Operating
101861    Base Salary/Compensation  NO_LABEL  PreK-12 Operating

                              Object_Description  \
344986                             EMPLOYEE BENEFITS
384803                             EMPLOYEE BENEFITS
224382                always OTHER PERSONAL SERVICES
305347  Extra Duty Pay/Overtime For Support Personnel
101861                  SALARIES OF REGULAR EMPLOYEES

                ...                 \
344986              ...
384803              ...
224382              ...
305347              ...
101861              ...

                          Sub_Object_Description  Location_Description  \
344986                                        NaN                   NaN
384803                                        NaN  PERSONNEL-PAID LEAVE
224382                                        NaN              School
305347  Extra Duty Pay/Overtime For Support Personnel           Unallocated
101861                                        NaN                   NaN

    FTE                    Function_Description Facility_or_Department  \
344986  NaN                UNALLOC BUDGETS/SCHOOLS                    NaN
384803  NaN                            NON-PROJECT                    NaN
224382  0.0         EXCEPTIONAL                                       NaN
305347  NaN  Facilities Maintenance And Operations    Gifted And Talented
101861  NaN                                TITLE I                    NaN

                Position_Extra       Total  \
344986   PROFESSIONAL-INSTRUCTIONAL    27.04000
384803   PROFESSIONAL-INSTRUCTIONAL         NaN
224382                          NaN   200.39000
305347  ANY CUS WHO IS NOT A SUPER      5.29000
101861   PROFESSIONAL-INSTRUCTIONAL  1575.03504

             Program_Description                Fund_Description  \
344986  GENERAL HIGH SCHOOL EDUCATION                             NaN
384803                 STAFF SERVICES                             NaN
224382                            NaN  GENERAL FUND
```

```
305347           Gifted And Talented          General Operating Fund
101861   GENERAL ELEMENTARY EDUCATION                            NaN

                          Text_1
344986            REGULAR INSTRUCTION
384803                        CENTRAL
224382                            NaN
305347  ADDL REGULAR PAY-NOT SMOOTHED
101861            REGULAR INSTRUCTION

[5 rows x 25 columns]
```

Which of these is a classification problem?

You'll continue your EDA in this exercise by computing summary statistics for the numeric data in the dataset. The data has been pre-loaded into a DataFrame called df.

You can use df.info() in the IPython Shell to determine which columns of the data are numeric, specifically type float64. You'll notice that there are two numeric columns, called FTE and Total.

- FTE: Stands for "full-time equivalent". If the budget item is associated to an employee, this number tells us the percentage of full-time that the employee works. A value of 1 means the associated employee works for the schooll full-time. A value close to 0 means the item is associated to a part-time or contracted employee.
- Total: Stands for the total cost of the expenditure. This number tells us how much the budget item cost.

After printing summary statistics for the numeric data, your job is to plot a histogram of the non-null FTE column to see the distribution of part-time and full-time employees in the dataset.

Instructions

- Print summary statistics of the numeric columns in the DataFrame df using the .describe() method.
- Import matplotlib.pyplot as plt.
- Create a histogram of the non-null 'FTE' column. You can do this by passing df['FTE'].dropna() to plt.hist().
- The title has been specified and axes have been labeled, so hit 'Submit Answer' to see how often school employees work full-time!

```python
# Print the summary statistics
print(df.describe())

# Import matplotlib.pyplot as plt
import matplotlib.pyplot as plt

# Create the histogram
plt.hist(df['FTE'].dropna())

# Add title and labels
plt.title('Distribution of %full-time \n employee works')
plt.xlabel('% of full-time')
plt.ylabel('num employees')

# Display the histogram
plt.show()
```

Results :

```
<script.py> output:
              FTE         Total
    count  449.000000  1.542000e+03
    mean     0.493532  1.446867e+04
    std      0.452844  7.916752e+04
    min     -0.002369 -1.044084e+06
    25%           NaN           NaN
    50%           NaN           NaN
    75%           NaN           NaN
    max      1.047222  1.367500e+06
```

see : img/graph1.svg

Nice! The high variance in expenditures makes sense (some purchases are cheap some are expensive). Also, it looks like the FTE column is bimodal. That is, there are some part-time and some full-time employees.

Exploring datatypes in pandas

It's always good to know what datatypes you're working with, especially when the inefficient pandas type object may be involved. Towards that end, let's explore what we have.

The data has been loaded into the workspace as df. Your job is to look at the DataFrame attribute .dtypes in the IPython Shell, and call its .value_counts() method in order to answer the question below.

Make sure to call df.dtypes.value_counts(), and not df.value_counts()! Check out the difference in the Shell. df.value_counts() will return an error, because it is a Series method, not a DataFrame method.

How many columns with dtype object are in the data?

Possible Answers : => 2

- 2.
- 23.
- 64.
- 25.

```
df.dtypes.value_counts()
```

Results :

```
In [1]: df.dtypes.value_counts()
Out[1]:
object     23
float64     2
dtype: int64
```

Encode the labels as categorical variables

Remember, your ultimate goal is to predict the probability that a certain label is attached to a budget line item. You just saw that many columns in your data are the inefficient object type. Does this include the labels you're trying to predict? Let's find out!

There are 9 columns of labels in the dataset. Each of these columns is a category that has many possible values it can take). The 9 labels have been loaded into a list called LABELS. In the Shell, check out the type for these labels using df[LABELS].dtypes.

You will notice that every label is encoded as an object datatype. Because category datatypes are much more efficient your task is to convert the labels to category types using the .astype() method.

Note: .astype() only works on a pandas Series. Since you are working with a pandas DataFrame, you'll need to use the .apply() method and provide a lambda function called categorize_label that applies .astype() to each column, x.

Instructions

- Define the lambda function categorize_label to convert column x into x.astype('category').
- Use the LABELS list provided to convert the subset of data df[LABELS] to categorical types using the .apply() method and categorize_label. Don't forget axis=0.
- Print the converted .dtypes attribute of df[LABELS].

```
# Define the lambda function: categorize_label
categorize_label = lambda x: x.astype('category')

print(LABELS)
print('------------')
print(df[LABELS].dtypes)
print('------------')

# Convert df[LABELS] to a categorical type
df[LABELS] = df[LABELS].apply(categorize_label,axis=0)

# Print the converted dtypes
print(df[LABELS].dtypes)
```

Results :

```
<script.py> output:
    ['Function', 'Use', 'Sharing', 'Reporting', 'Student_Type', 'Position_Type', 'Object_Type', '
Pre_K', 'Operating_Status']
    ------------
    Function            object
    Use                 object
    Sharing             object
    Reporting           object
    Student_Type        object
    Position_Type       object
    Object_Type         object
    Pre_K               object
    Operating_Status    object
    dtype: object
    ------------
    Function            category
    Use                 category
    Sharing             category
    Reporting           category
    Student_Type        category
    Position_Type       category
    Object_Type         category
```

```
Pre_K                category
Operating_Status     category
dtype: object
```

Great work! You're getting close to something you can work with. Keep it up!

Which of these is a classification problem?

As Peter mentioned in the video, there are over 100 unique labels. In this exercise, you will explore this fact by counting and plotting the number of unique values for each category of label.

The dataframe df and the LABELS list have been loaded into the workspace; the LABELS columns of df have been converted to category types.

pandas, which has been pre-imported as pd, provides a pd.Series.nunique method for counting the number of unique values in a Series.

Instructions

- Create the DataFrame num_unique_labels by using the .apply() method on df[LABELS] with pd.Series.nunique as the argument.
- Create a bar plot of num_unique_labels using pandas' .plot(kind='bar') method.
- The axes have been labeled for you, so hit 'Submit Answer' to see the number of unique values for each label.

```python
# Import matplotlib.pyplot
import matplotlib.pyplot as plt

# Calculate number of unique values for each label: num_unique_labels
num_unique_labels = df[LABELS].apply(pd.Series.nunique)

# Plot number of unique values for each label
num_unique_labels.plot(kind='bar')

# Label the axes
plt.xlabel('Labels')
plt.ylabel('Number of unique values')

# Display the plot
plt.show()
```

Results :

Woah! That's a lot of labels to work with. How will you measure success with these many labels? You'll find out in the next video!

Penalizing highly confident wrong answers

As Peter explained in the video, log loss provides a steep penalty for predictions that are both wrong and confident, i.e., a high probability is assigned to the incorrect class.

Suppose you have the following 3 examples:

- A:y=1,p=0.85
- B:y=0,p=0.99
- C:y=0,p=0.51

Select the ordering of the examples which corresponds to the lowest to highest log loss scores. y is an indicator of whether the example was classified correctly. You shouldn't need to crunch any numbers!

Possible Answers : => 3

- Lowest: A, Middle: B, Highest: C.
- Lowest: C, Middle: A, Highest: B.
- Lowest: A, Middle: C, Highest: B.
- Lowest: B, Middle: A, Highest: C.

```

```

Results :

Yes! Of the two incorrect predictions, B will have a higher log loss because it is confident and wrong.

Computing log loss with NumPy

To see how the log loss metric handles the trade-off between accuracy and confidence, we will use some sample data generated with NumPy and compute the log loss using the provided function compute_log_loss(), which Peter showed you in the video.

5 one-dimensional numeric arrays simulating different types of predictions have been pre-loaded: actual_labels, correct_confident, correct_not_confident, wrong_not_confident, and wrong_confident.

Your job is to compute the log loss for each sample set provided using the compute_log_loss(predicted_values, actual_values). It takes the predicted values as the first argument and the actual values as the second argument.

Instructions

- Using the compute_log_loss() function, compute the log loss for the following predicted values (in each case, the actual values are contained in actual_labels):
  - correct_confident.
  - correct_not_confident.
  - wrong_not_confident.
  - wrong_confident.
  - actual_labels.

```
# Compute and print log loss for 1st case
correct_confident = compute_log_loss(correct_confident, actual_labels)
print("Log loss, correct and confident: {}".format(correct_confident))

# Compute log loss for 2nd case
correct_not_confident = compute_log_loss(correct_not_confident, actual_labels)
print("Log loss, correct and not confident: {}".format(correct_not_confident))

# Compute and print log loss for 3rd case
wrong_not_confident = compute_log_loss(wrong_not_confident, actual_labels)
```

```
print("Log loss, wrong and not confident: {}".format(wrong_not_confident))

# Compute and print log loss for 4th case
wrong_confident = compute_log_loss(wrong_confident, actual_labels)
print("Log loss, wrong and confident: {}".format(wrong_confident))

# Compute and print log loss for actual labels
actual_labels = compute_log_loss(actual_labels, actual_labels)
print("Log loss, actual labels: {}".format(actual_labels))
```

Results :

```
In [1]: actual_labels
Out[1]: array([ 1.,  1.,  1.,  1.,  1.,  0.,  0.,  0.,  0.,  0.])

In [2]: correct_confident
Out[2]: array([ 0.95,  0.95,  0.95,  0.95,  0.95,  0.05,  0.05,  0.05,  0.05,  0.05])

<script.py> output:
    Log loss, correct and confident: 0.05129329438755058
    Log loss, correct and not confident: 0.4307829160924542
    Log loss, wrong and not confident: 1.049822124498678
    Log loss, wrong and confident: 2.9957322735539904
    Log loss, actual labels: 9.99200722162646e-15
```

Wow! Log loss penalizes highly confident wrong answers much more than any other type. This will be a good metric to use on your models. You rock!

Which of these is a classification problem?

Possible Answers : => 1

-

```

```

Results :

Which of these is a classification problem?

Possible Answers : => 1

-

```

```

Results :

Which of these is a classification problem?

Possible Answers : => 1

-

Results :

Which of these is a classification problem?

Possible Answers : => 1

-

Results :

Which of these is a classification problem?

Possible Answers : => 1

-

Results :

Which of these is a classification problem?

Possible Answers : => 1

-

Results :

Which of these is a classification problem?

Possible Answers : => 1

-

Results :

Which of these is a classification problem?

Possible Answers : => 1

-

Results :

Which of these is a classification problem?

Possible Answers : => 1

-

Results :

Which of these is a classification problem?

Possible Answers : => 1

-

Results :

Which of these is a classification problem?

Possible Answers : => 1

-

Results :

Which of these is a classification problem?

Possible Answers : => 1

•

Results :

Which of these is a classification problem?

Possible Answers : => 1

•

Results :

Which of these is a classification problem?

Possible Answers : => 1

•

Results :

Which of these is a classification problem?

Possible Answers : => 1

•

Results :

Which of these is a classification problem?

Possible Answers : => 1

•

Results :

Which of these is a classification problem?

Possible Answers : => 1

-

Results :

Which of these is a classification problem?

Possible Answers : => 1

-

Results :

Which of these is a classification problem?

Possible Answers : => 1

-

Results :

Which of these is a classification problem?

Possible Answers : => 1

-

Results :

Which of these is a classification problem?

Possible Answers : => 1

-

Results :