assignment4

October 24, 2023

---

*You are currently looking at **version 1.0** of this notebook. To download notebooks and datafiles, as well as get help on Jupyter notebooks in the Coursera platform, visit the Jupyter Notebook FAQ course resource.*

---

1 Assignment 4 - Predicting and understanding viewer engagement with educational videos

With the accelerating popularity of online educational experiences, the role of online lectures and other educational video continues to increase in scope and importance. Open access educational repositories such as videolectures.net, as well as Massive Open Online Courses (MOOCs) on platforms like Coursera, have made access to many thousands of lectures and tutorials an accessible option for millions of people around the world. Yet this impressive volume of content has also led to a challenge in how to find, filter, and match these videos with learners. This assignment gives you an example of how machine learning can be used to address part of that challenge.

1.1 About the prediction problem

One critical property of a video is engagement: how interesting or "engaging" it is for viewers, so that they decide to keep watching. Engagement is critical for learning, whether the instruction is coming from a video or any other source. There are many ways to define engagement with video, but one common approach is to estimate it by measuring how much of the video a user watches. If the video is not interesting and does not engage a viewer, they will typically abandon it quickly, e.g. only watch 5 or 10% of the total.

A first step towards providing the best-matching educational content is to understand which features of educational material make it engaging for learners in general. This is where predictive modeling can be applied, via supervised machine learning. For this assignment, your task is to predict how engaging an educational video is likely to be for viewers, based on a set of features extracted from the video's transcript, audio track, hosting site, and other sources.

We chose this prediction problem for several reasons:

- It combines a variety of features derived from a rich set of resources connected to the original data;
- The manageable dataset size means the dataset and supervised models for it can be easily explored on a wide variety of computing platforms;

- Predicting popularity or engagement for a media item, especially combined with understanding which features contribute to its success with viewers, is a fun problem but also a practical representative application of machine learning in a number of business and educational sectors.

1.2 About the dataset

We extracted training and test datasets of educational video features from the VLE Dataset put together by researcher Sahan Bulathwela at University College London.

We provide you with two data files for use in training and validating your models: train.csv and test.csv. Each row in these two files corresponds to a single educational video, and includes information about diverse properties of the video content as described further below. The target variable is `engagement` which was defined as True if the median percentage of the video watched across all viewers was at least 30%, and False otherwise.

Note: Any extra variables that may be included in the training set are simply for your interest if you want an additional source of data for visualization, or to enable unsupervised and semi-supervised approaches. However, they are not included in the test set and thus cannot be used for prediction. **Only the data already included in your Coursera directory can be used for training the model for this assignment.**

For this final assignment, you will bring together what you've learned across all four weeks of this course, by exploring different prediction models for this new dataset. In addition, we encourage you to apply what you've learned about model selection to do hyperparameter tuning using training/validation splits of the training data, to optimize the model and further increase its performance. In addition to a basic evaluation of model accuracy, we've also provided a utility function to visualize which features are most and least contributing to the overall model performance.

File descriptions assets/train.csv - the training set (Use only this data for training your model!) assets/test.csv - the test set

Data fields

train.csv & test.csv:

`title_word_count - the number of words in the title of the video.`

`document_entropy - a score indicating how varied the topics are covered in the video, based on`

`freshness - The number of days elapsed between 01/01/1970 and the lecture published date. Vide`

`easiness - A text difficulty measure applied to the transcript. A lower score indicates more c`

`fraction_stopword_presence - A stopword is a very common word like 'the' or 'and'. This feature`

`speaker_speed - The average speaking rate in words per minute of the presenter in the video.`

`silent_period_rate - The fraction of time in the lecture video that is silence (no speaking).`

train.csv only:

`engagement - Target label for training. True if learners watched a substantial portion of the v`

1.3 Evaluation

Your predictions will be given as the probability that the corresponding video will be engaging to learners.

The evaluation metric for this assignment is the Area Under the ROC Curve (AUC).

Your grade will be based on the AUC score computed for your classifier. A model with an AUC (area under ROC curve) of at least 0.8 passes this assignment, and over 0.85 will receive full points.

For this assignment, create a function that trains a model to predict significant learner engagement with a video using `asset/train.csv`. Using this model, return a Pandas Series object of length 2309 with the data being the probability that each corresponding video from `readonly/test.csv` will be engaging (according to a model learned from the 'engagement' label in the training set), and the video index being in the `id` field.

Example:

```
id
   9240    0.401958
   9241    0.105928
   9242    0.018572
             ...
   9243    0.208567
   9244    0.818759
   9245    0.018528
         ...
   Name: engagement, dtype: float32
```

1.3.1 Hints

- Make sure your code is working before submitting it to the autograder.
- Print out and check your result to see whether there is anything weird (e.g., all probabilities are the same).
- Generally the total runtime should be less than 10 mins.
- Try to avoid global variables. If you have other functions besides engagement_model, you should move those functions inside the scope of engagement_model.
- Be sure to first check the pinned threads in Week 4's discussion forum if you run into a problem you can't figure out.

1.3.2 Extensions

- If this prediction task motivates you to explore further, you can find more details here on the original VLE dataset and others related to video engagement: https://github.com/sahanbull/VLE-Dataset

[1]:
```
import warnings
warnings.filterwarnings("ignore")

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

np.random.seed(0)   # Do not change this value: required to be compatible with␣
 ↪solutions generated by the autograder.
```

[2]:
```
train = pd.read_csv('assets/train.csv')
```

[3]:
```
train.head()
```

[3]:

	id	title_word_count	document_entropy	freshness	easiness	fraction_stopword_presence	normalization_rate	speaker_speed	silent_period_rate	engagement
0	1	9	7.753995	16310	75.583936	0.553664	0.034049	2.997753	0.0	True
1	2	6	8.305269	15410	86.870523	0.584498	0.018763	2.635789	0.0	False
2	3	3	7.965583	15680	81.915968	0.605685	0.030720	2.538095	0.0	False
3	4	9	8.142877	15610	80.148937	0.593664	0.016873	2.259055	0.0	False
4	5	9	8.161250	14920	76.907549	0.581637	0.023412	2.420000	0.0	False

[4]:
```
test = pd.read_csv('assets/test.csv')
```

[5]:
```
train = train.set_index('id')
```

[6]:
```
test = test.set_index('id')
```

[7]:
```
X_train = train.iloc[:, :-1]
```

[8]:
```
y_train = train.iloc[:, -1]
```

[9]:
```
X_test = test
```

[51]:
```
X_test.index
```

[51]:
```
Int64Index([ 9240,  9241,  9242,  9243,  9244,  9245,  9246,  9247,  9248,
             9249,
            …
            11539, 11540, 11541, 11542, 11543, 11544, 11545, 11546, 11547,
            11548],
           dtype='int64', name='id', length=2309)
```

[10]:
```
from sklearn.svm import SVC

classifier_svc = SVC(probability = True)

classifier_svc.fit(X_train, y_train)
```

[10]:
```
SVC(probability=True)
```

[11]:
```
predictions_svc = classifier_svc.predict(X_test)
```

[12]:
```
predictions_prob_svc = classifier_svc.predict_proba(X_test)
```

[13]:
```
pd.set_option('display.max_rows', None)  # or 1000
```

[21]:
```
predictions_svc[:50]
```

[21]:
```
array([False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False])
```

[19]:
```
True in predictions_svc
```

[19]:
```
False
```

[73]:
```
predictions_prob_svc[:50]
```

[73]:
```
array([[0.81740085, 0.18259915],
       [0.81330443, 0.18669557],
       [0.93792716, 0.06207284],
       [0.88389516, 0.11610484],
       [0.8194043 , 0.1805957 ],
       [0.81483021, 0.18516979],
       [0.81460689, 0.18539311],
       [0.81789269, 0.18210731],
       [0.81663314, 0.18336686],
```

```
       [0.81572905, 0.18427095],
       [0.81354481, 0.18645519],
       [0.82016927, 0.17983073],
       [0.82396885, 0.17603115],
       [0.8206651 , 0.1793349 ],
       [0.85359845, 0.14640155],
       [0.83839422, 0.16160578],
       [0.82343497, 0.17656503],
       [0.81339666, 0.18660334],
       [0.813039  , 0.186961  ],
       [0.85493476, 0.14506524],
       [0.82636665, 0.17363335],
       [0.81953963, 0.18046037],
       [0.81697165, 0.18302835],
       [0.93560705, 0.06439295],
       [0.81670278, 0.18329722],
       [0.87680929, 0.12319071],
       [0.82407282, 0.17592718],
       [0.81875347, 0.18124653],
       [0.82312681, 0.17687319],
       [0.82453987, 0.17546013],
       [0.88130198, 0.11869802],
       [0.84368802, 0.15631198],
       [0.82080851, 0.17919149],
       [0.81542365, 0.18457635],
       [0.93807911, 0.06192089],
       [0.95740138, 0.04259862],
       [0.86407031, 0.13592969],
       [0.9403202 , 0.0596798 ],
       [0.87836254, 0.12163746],
       [0.82809947, 0.17190053],
       [0.87694712, 0.12305288],
       [0.81763113, 0.18236887],
       [0.81475694, 0.18524306],
       [0.82709303, 0.17290697],
       [0.88819297, 0.11180703],
       [0.82613631, 0.17386369],
       [0.81846807, 0.18153193],
       [0.85500755, 0.14499245],
       [0.81711485, 0.18288515],
       [0.81860318, 0.18139682]])
```

[72]:
```
predictions_prob_svc[:, 1]
```

[72]:
```
array([0.18259915, 0.18669557, 0.06207284, …, 0.18444146, 0.14236555,
       0.18702427])
```

[22]:
```
from sklearn.ensemble import RandomForestClassifier

classifier_random_forest = RandomForestClassifier(random_state=42)

classifier_random_forest.fit(X_train, y_train)
```

[22]:
```
RandomForestClassifier(random_state=42)
```

[23]:
```
predictions_random_forest = classifier_random_forest.predict(X_test)
```

[24]:
```
predictions_prob_random_forest = classifier_random_forest.predict_proba(X_test)
```

[25]:
```
predictions_random_forest[:100]
```

[25]:
```
array([False, False, False,  True, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False,  True, False, False,
       False, False, False, False, False,  True, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False])
```

[26]:
```
True in predictions_random_forest
```

[26]:
```
True
```

[30]:
```
predictions_prob_random_forest[:50]
```

[30]:
```
array([[1.  , 0.  ],
       [0.96, 0.04],
       [0.84, 0.16],
       [0.13, 0.87],
       [0.98, 0.02],
       [0.98, 0.02],
       [0.98, 0.02],
       [0.99, 0.01],
       [0.88, 0.12],
       [0.99, 0.01],
       [1.  , 0.  ],
       [0.98, 0.02],
       [1.  , 0.  ],
       [1.  , 0.  ],
```

```
       [0.97, 0.03],
       [0.97, 0.03],
       [0.97, 0.03],
       [0.99, 0.01],
       [0.97, 0.03],
       [1.  , 0.  ],
       [0.95, 0.05],
       [0.99, 0.01],
       [0.94, 0.06],
       [0.84, 0.16],
       [1.  , 0.  ],
       [0.62, 0.38],
       [0.99, 0.01],
       [0.99, 0.01],
       [0.97, 0.03],
       [1.  , 0.  ],
       [0.97, 0.03],
       [0.63, 0.37],
       [0.99, 0.01],
       [0.94, 0.06],
       [0.87, 0.13],
       [0.79, 0.21],
       [0.99, 0.01],
       [0.73, 0.27],
       [0.66, 0.34],
       [0.99, 0.01],
       [1.  , 0.  ],
       [0.99, 0.01],
       [0.97, 0.03],
       [0.97, 0.03],
       [0.8 , 0.2 ],
       [0.99, 0.01],
       [0.77, 0.23],
       [0.96, 0.04],
       [1.  , 0.  ],
       [0.97, 0.03]])
```

[28]:
```
predictions_prob_random_forest[:, 1]
```

[28]:
```
array([0.  , 0.04, 0.16, …, 0.  , 0.89, 0.  ])
```

[53]:
```
test_index = X_test.index
```

[55]:
```
predictions_prob_random_forest = predictions_prob_random_forest[:, 1]
```

[57]:
```
predictions_prob_random_forest = pd.Series(predictions_prob_random_forest)
```

[61]:
```
pd.set_option('display.max_rows', 100)  # or 1000
```

[64]:
```
predictions_prob_random_forest = predictions_prob_random_forest.
↪set_axis(test_index)
```

[65]:
```
predictions_prob_random_forest
```

[65]:
```
id
9240     0.00
9241     0.04
9242     0.16
9243     0.87
9244     0.02
         …
11544    0.01
11545    0.00
11546    0.00
11547    0.89
11548    0.00
Length: 2309, dtype: float64
```

[ ]:

[ ]:

[70]:
```
def engagement_model():

    rec = None

    # YOUR CODE HERE

    from sklearn.ensemble import RandomForestClassifier

    classifier_random_forest = RandomForestClassifier(random_state=42)

    classifier_random_forest.fit(X_train, y_train)

    predictions_prob_random_forest = classifier_random_forest.
↪predict_proba(X_test)

    predictions_prob_random_forest = predictions_prob_random_forest[:, 1]

    rec = predictions_prob_random_forest

    rec = pd.Series(rec)

    test_index = X_test.index
```

```
    rec = rec.set_axis(test_index)

    return rec

    raise NotImplementedError()
```

[68]:
```
stu_ans = engagement_model()
assert isinstance(stu_ans, pd.Series), "Your function should return a pd.Series.
 ↪ "
assert len(stu_ans) == 2309, "Your series is of incorrect length: expected 2309␣
 ↪"
assert np.issubdtype(stu_ans.index.dtype, np.integer), "Your answer pd.Series␣
 ↪should have an index of integer type representing video id."
```

[ ]:

[ ]:

[69]:
```
engagement_model().head(25)
```

[69]:
```
id
9240    0.00
9241    0.04
9242    0.16
9243    0.87
9244    0.02
9245    0.02
9246    0.02
9247    0.01
9248    0.12
9249    0.01
9250    0.00
9251    0.02
9252    0.00
9253    0.00
9254    0.03
9255    0.03
9256    0.03
9257    0.01
9258    0.03
9259    0.00
9260    0.05
9261    0.01
9262    0.06
9263    0.16
9264    0.00
```

```
dtype: float64
```