Toggle navigation Udacity Logo

- Logout
- Logout

Return to "Machine Learning Engineer Nanodegree" in the classroom

Finding Donors for CharityML

- Review
- Code Review
- History

Meets Specifications

:trophy: awesome job on this submission! You show a great amount of understanding of supervised learning and your ability to implement the models it was clearly reflected in your work.

Exploring the Data

Student's implementation correctly calculates the following:

- Number of records
- Number of individuals with income >$50,000
- Number of individuals with income <=$50,000
- Percentage of individuals with income > $50,000

great work you have properly calculated all dataset stats ! :sunglasses:

- [x] number of records : 45222
- [x] number of individuals with income >$50,000 : 11208
- [x] number of individuals with income <=$50,000 : 34014
- [x] percentage of individuals with income > $50,000 : 24.78%

:bulb: note 1:

> the finality of this exercise is not only to find the proper values but also to get you familiar with a tool that will be your bread and butter on this field `pandas` so let's see some other ways to use this tool:

shape

as explained here: shape and even better here: numpy.shape shape returns the a tuple with the dimensions of a n-dimensional array.

so using this on this dataset we would have the following:

- `shape[0]` = to the number of rows
- `shape[1]` = to the number of columns

in resume you could use this for example:

```
n_greater_50k = data[data['income']=='>50k'].shape[0]
```

value counts

pandas offer many tools and count is really useful her when combined with the python abilities to unpack returns we could simplify the code like this:

```
n_at_most_50k, n_greater_50k = data.income.value_counts()
```

:bulb: note 2:

> as is noticeable our data is imbalanced that is really common problem in the field.

here are some articles on how to tackle this problem:

- https://www.kdnuggets.com/2017/06/7-techniques-handle-imbalanced-data.html
- https://towardsdatascience.com/what-metrics-should-we-use-on-imbalanced-data-set-precision-recall-roc-e2e79252aeba
- http://www.pitt.edu/~jeffcohn/biblio/jeni_metrics.pdf

:bulb: note 3:

have a look on this site: https://pair-code.github.io/facets/
you can have nice insight on the data by using nice plots like:

tmp.png

Preparing the Data

Student correctly implements one-hot encoding for the feature and income data.

Great Job on encoding the features and target labels using `get_dummies`.
:bulb: Note:

> it is also possible to use Label Encoder as an alternative, especially when we are dealing with a Multi Class prediction case.

here is an example on how to use LabelEncoder:

```
encoder = LabelEncoder()
income = encoder.fit_transform(income_raw)
```

or you can use Numpy another tool that will be party of your toolkit:
here is how to make a one-hot encoding with Numpy

```
import numpy as np
nb_classes = 6
targets = np.array([[2, 3, 4, 0]]).reshape(-1)
one_hot_targets = np.eye(nb_classes)[targets]
```

what returns:

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

here are some interesting articles about the subject:

http://pbpython.com/categorical-encoding.html
https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/

Evaluating Model Performance

Student correctly calculates the benchmark score of the naive predictor for both accuracy and F1 scores.

Great work on correctly calculating the Accuracy, Precision and Recall!

Also, well done getting the right calculation of F-score.

:bulb: Note:

> with `TN and FN = 0` the Accuracy and Precision don't differ!

and here an article worth reading about metrics

https://medium.com/greyatom/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b

The pros and cons or application for each model is provided with reasonable justification why each model was chosen to be explored.

Please list all the references you use while listing out your pros and cons.

nice discussion. you show a great level of understanding of the models.

:bulb: note:

> to know the pros and cons of a model is really important as this will help you select the best model based on the data-set. and is one of the main questions you are going to face when interviewing for machine learning roles, have a look at the following articles about this.

comprehensive sas article on model selection: https://blogs.sas.com/content/subconsciousmusings/2017/04/12/machine-learning-algorithm-use/

great quora thread comparing common ml models: https://www.quora.com/what-are-the-advantages-of-different-classification-algorithms

and also here is a cheat-sheet for this:
ml_map.png

Student successfully implements a pipeline in code that will train and predict on the supervised learning algorithm given.

Nice Job on properly implementing the pipeline :sunglasses:
:bulb: Note:
Most of the time when working with ML you will be implementing a pipeline for training and prediction.
And well define those stages is really important and fun.

also `sklearn's` have a utility to help with that if you feel curious here is the link to it Pipeline

Student correctly implements three supervised learning models and produces a performance visualization.

Improving Results

Justification is provided for which model appears to be the best to use given computational cost, model performance, and the characteristics of the data.

Student is able to clearly and concisely describe how the optimal model works in layman's terms to someone who is not familiar with machine learning nor has a technical background.

Really nice :+1:
You showed that you really understand the model.
:bulb: note:
When trying to explain an algorithm to a non-technical person is really good to try to use some examples (like creating a scenario and running step by step of the algorithm) and avoiding the use of technical terms as it can get cumbersome really quickly. when you can't avoid the use of a technical term focus on explaining what this term means.

another good approach is to use images that make your explanation rich and easy to understand.

The final model chosen is correctly tuned using grid search with at least one parameter using at least three settings. If the model does not need any parameter tuning it is explicitly stated with reasonable justification.

Student reports the accuracy and F1 score of the optimized, unoptimized, models correctly in the table provided. Student compares the final model results to previous results obtained.

Feature Importance

Student ranks five features which they believe to be the most relevant for predicting an individual's' income. Discussion is provided for why these features were chosen.

Student correctly implements a supervised learning model that makes use of the `feature_importances_` attribute. Additionally, student discusses the differences or similarities between the features they considered relevant and the reported relevant features.

Nice work, it is worth mention that those are the `feature_importances_` for GradientBoostingClassifier and it may vary depending on the model used.
For example, if the Random Forest is used here, the following is returned:
randomf.png

Student analyzes the final model's performance when only the top 5 features are used and compares this performance to the optimized model from **Question 5**.

Really good work :clap:

:bulb: Note:

> reducing the number of features on a model is really important to avoid the Curse of Dimensionality and the secret resides on finding the proper number of features that better fits between time and performance.

Download Project
Return to Path
Rate this review
( ) ( ) ( ) ( ) ( )
Rate this review
( ) ( ) ( ) ( ) ( )