

## PROJECT

## Finding Donors for CharityML

A part of the Machine Learning Engineer Nanodegree Program

## PROJECT REVIEW


## NOTES


SHARE YOUR ACCOMPLISHMENT!  

## Meets Specifications

Awesome submission I had a really great time reviewing your work 

You have shown that you understand the supervised methodologies of ML in depth and are now ready to tackle more challenging topics this nano-degree can offer!

I would like to personally congratulate on completing this project which I personally feel like a stepping stone to the world of ML. 

Keep up the good work buddy and all the best on the rest of your learning 

## 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

Good Job getting the dataset stats ! 

You have correctly calculated:

- Number of records : 45222
- Number of individuals with income >\$50,000 : 11208
- Number of individuals with income <=\$50,000 : 34014
- Percentage of individuals with income > \$50,000 : 24.78%

You can also use the `shape` method to get your label counts.

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

**Note:** In case when the Target Class is imbalanced (same as in our current scenario)

From the dataset stats we can see that the dataset we have is an **Imbalanced** dataset! Which means that the ratio of individuals making more than \$50k vs those making less are not in proportion, and in such cases we should make sure that the metric we will be using for model evaluation can capture how well the model is actually doing. (Additional info [here](#))

As you can see, In this project we are using the precision, recall, and F-beta scores( `beta` being 0.5). Alternatively we can also consider F1 score (which is equivalent to using F-beta with `beta = 1` ).

**F1 Score:**

Often referred to as the traditional F-measure or balanced F-score (F1 score) is the harmonic mean of precision and recall

$$F_1 = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

## 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` .

Your code outputs total 103 features after One Hot Encoding, which is the expected value.

Something to note here is that we can also use Label Encoder as an alternative in case where we have a huge number of output classes **Multi Class** predictions. **Label Encoder** can be implemented as below:

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

Also please go through the below article which explains when and why we use OHE

[Link](#)

For your reference, check out this [post](#) that demonstrates various ways in which we can handle categorical variables. 😎

## Evaluating Model Performance

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

Great work on correctly expressing Accuracy, Precision and Recall! Also, well done getting the right expression for the F-score. 👍

- Both accuracy and F-score are correctly calculated: Accuracy score: 0.2478 , F-Score: 0.2917

💡 Note:

- As  $TN = 0$  and  $FN = 0$ ; both the Accuracy and Precision in our case is the same!
- Also check out [this](#) nice article which explains performance metrics for Classification Problems!

**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.**

Pros and Cons of the models:

Nice discussion. Seems like you really understood the models.

As a side-note, please check out this [Quora](#) post which will help while selecting a model from a bunch of options and which model is suitable for what kind of a problem.

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

Nice job building this pipeline! Machine learning pipelines are very useful tools, when you get to larger-scale ML projects you'll often find that you'll perform experiments to see which models (and what hyperparameters) perform best. When you define the pipeline of the entire model process (from preprocessing to prediction), it's easier to conduct repeatable experiments.

[Here](#) you can check out sklearn's Pipeline implementation for a more robust version of what we've built here.

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

Great Implementation here! 😎

Note:

If you notice the visualization closely, there are no F Score representation for SVC in case of both Training and Prediction for 1% of the data. And a `warning` is displayed as:

```
UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 due to no predicted samples.
```

Reason:

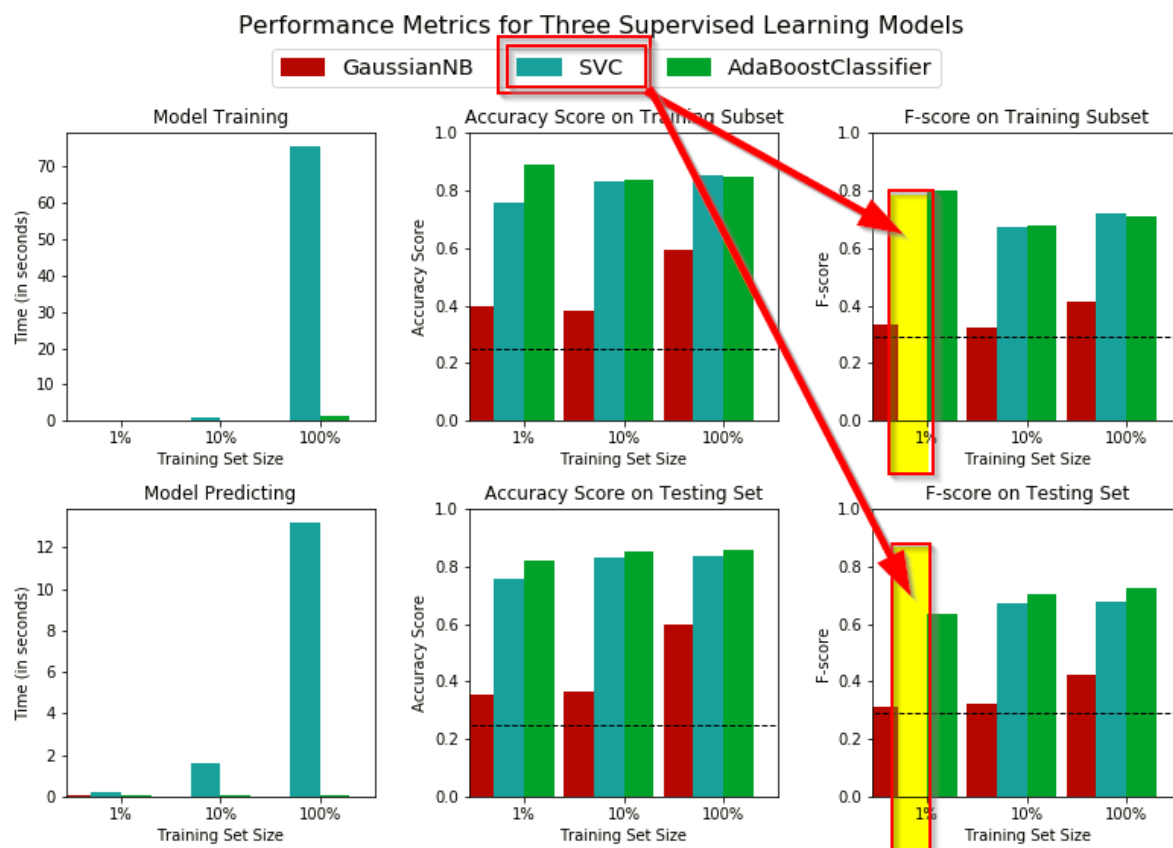
As you know from earlier implementation:

- Precision =  $TP/(TP+FP)$  : from the warning, predictor doesn't predicts positive class at all - precision is 0.
- Recall =  $TP/(TP+FN)$ , in case if predictor doesn't predict positive class - TP is 0 - recall is 0.

So now we are dividing 0/0. So F-Score shows *ill-defined*

Here, for 1% of the data, the data subset is so small that SVC doesnot output any postive predicted value and so

`TP = 0`



## 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.

Agree with your model selection! 👍

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.

Great discussion here! 👍

You have put in a lot of effort to explain the model as simply as possible. 🙌

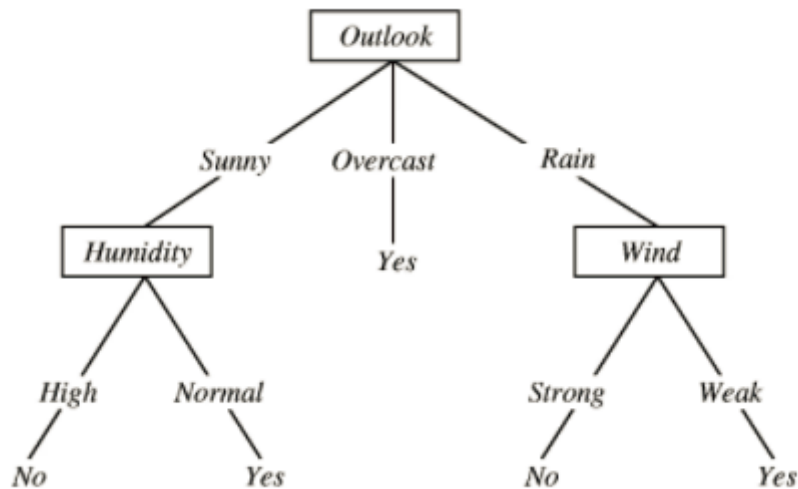


**Note:** For your reference please go through the Layman's explanation of a **Random Forrest Classifier** and a **Logistic Regression Model**:

### The Random Forest Classifier

*Random Forest Classifier* is considered to be an ensemble method type of model. Ensemble methods basically gather together the results of multiple trained models to create a much better model. In the case of the Random Forest Classifier, it takes the results of multiple decision tree models, uses their individual predictions to "vote" for one, much more informed prediction. For example, if there are five decision trees in our Random Forest model, and three of them predict a label of 1 and two of them predict a label of 0, our Random Forest model will return a prediction of 1.

To better understand the underlying nature of the Random Forest Classifier, we must also understand the Decision Tree Classifier. A decision tree can be thought of like a flow chart built of nodes, branches, and leaves. At each node, we ask a question of the data. Ex. What is the outlook?. The branch represents the possible answers, in this case, sunny, overcast, or rain. At the end of the branch is a leaf which represents the classification label. In this example, if it is overcast, then yes, the event will happen. Decision Trees can be made of multiple levels, a variable we call depth. Below is an example of one such decision tree graph.



In this example, based on features about the weather, we are trying to predict whether an event will happen. Now, imagine building this model multiple times with random samples from our dataset to give us a more informed prediction.

#### Training and predicting:

So how does a decision tree learn? How does the model know where to split the data at a node into separate branches? Well, remember that the goal of a decision tree is to classify a record with a certain label. And the goal of each split in the tree is to separate the data as much as possible into the possible labels. The ideal split for example would be for the left branch to only contain Class A labels and the right branch to only contain Class B labels. The tree measures the effectiveness of the split by using something called a loss function, which could either be the gini index or entropy. The model will go through the whole dataset and calculate the loss function for all possible splits, to find the one that describes the data the best. It would then use that particular split and increase the depth one level. Then it would calculate the loss function again for each node at that level and make the appropriate splits until the max depth of the tree is reached.

When making a prediction, the data is fed into the tree and runs through the trained model using the previously determined split thresholds until it ends at a leaf, resulting in the predicted label.

And remember, with Random Forest models, this happens with many trees and each tree results in a "vote" for the label. Majority vote wins!

---

## Logistic Regression:

Considering Logistic Regression to be the best fit for our problem, let us try to understand what regression is and then let us try to explain what Logistic Regression is (In terms of Statistics/Machine Learning). Regression is, in the simplest terms, understanding the relationship between the input and the output variables. This linear regression is something that gives a linear relationship between the input and output variables.

Logistic regression is basically a linear regression model which explains the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables with only difference being, the output for linear regression is a number that has its real meaning (a label) while the output of a logistic regression is a number that represents the probability of the event happening (i.e. the probability of people clicking an ad online, the probability of death in titanic disaster, etc.). In our case predicting the probability of an individual making more money than \$50,000. The intuition behind logistic regression is to transform the output of a linear regression to a range of (0,1) in which the probability lies. So, if for a person we

are trying to find if he likes apple or mango, Linear regression will give an output like A or B (A: person likes Apple, B: person likes Mango) while logistic regression will output probabilities for both the A and B. So, if  $(\text{Probability})A > \text{Probability}(B)$ , the person likes Apple.

Now as we understood what Logistic regression is and what it does, let us try to understand how our logistic Regression Model works. Our purpose is to classify a person as earning more than 50k or earning less than 50k. For our task, we have some data which has certain features like Age, Education Level, Hours per Week. So, from basic understanding we would say that a person having higher education who works more hours per week would earn more. And someone with same education working same hours, but younger will earn less because he has less work experience. Our model also works the same way, but with huge no of features and huge population of data. It analyzes all the features and figures our relationship between the features. It finds out which features greatly impacts a person's earning and which does not. So, after the model is done with figuring out the training data, it defines a boundary. This process is called training the Model. After training we are ready to predict if a person earns > 50 k or not. In this process, the model will input the data features to the already trained model and output probabilities belonging to each group. The person will belong to the class whose probability is greater.

Also I have added a few links which explains different ML models in simple terms. Go through each and it will help you improve your understanding on them.

<https://www.quora.com/What-is-logistic-regression>

<https://rayli.net/blog/data/top-10-data-mining-algorithms-in-plain-english/>

<http://xgboost.readthedocs.io/en/latest/model.html>

**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.**

Successfully Implemented:

- `GridSearchCV` on one parameter with at-least 3 values !

Great Job 🙌

**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.**

Logical Selection of features and the justification intuitively makes sense ! 👍

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.

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.

Great work!



Please note that often more than not, in instances where we really value training time or training time is a crucial deciding factor for model selection, we can also opt for a simpler model. Because cutting down on `features` on complex models often result in worse performance and worse training time.

 [DOWNLOAD PROJECT](#)

[RETURN TO PATH](#)

---

[Student FAQ](#)