UDACITY

< Back to Machine Learning Engineer Nanodegree

Machine Learning Capstone Project

REVIEW

CODE REVIEW

HISTORY

Requires Changes

3 SPECIFICATIONS REQUIRE CHANGES

This is a very cool analysis and a great read to a very real world and practical problem. You have demonstrated a full understanding of the entire machine learning pipeline and your report definitely gets the readers attention with the results you have achieved. You just have to expand in a few of these sections, but will greatly improve your report. Please check out some of these other ideas presented here and we look forward in seeing your next submission!!

Definition

Student provides a high-level overview of the project in layman's terms. Background information such as the problem domain, the project origin, and related data sets or input data is given.

Nice work here with your opening section, as you have given good starting paragraphs to outline the project and have provided background information on the problem domain. Definitely a real world problem.

And you have provided good research to back your claims. It is always important to provide similiar research on such a topic.

The problem which needs to be solved is clearly defined. A strategy for solving the problem, including discussion of the expected solution, has been made.

> "As part of this project, we developed a supervised machine learning model to predict customer churn from a given set of customer metadata"

Problem statement is clearly defined here. Would also recommend explicitly mentioning that this would be a classification problem in this section.

And very nice job mentioning your machine learning pipeline here, as this gives the reader some ideas in what is to come in your report and how you plan on solving this important task.

Metrics used to measure performance of a model or result are clearly defined. Metrics are justified based on the characteristics of the problem.

> "For this analysis, the primary metric that was used to determine the quality of both the baseline and the final model is the Recall score of the minor class (i.e., Class 1). "

Excellent reasoning for focusing on recall. Always important to tie your metric choice into your particular problem domain. A formula would also be a nice touch.

Glad that you also mention the unbalanced dataset.
(https://www.youtube.com/watch?v=2akd6uwtowc&list=PL6397E4B26D00A269&index=30)

Analysis

If a dataset is present, features and calculated statistics relevant to the problem have been reported and discussed, along with a sampling of the data. In lieu of a dataset, a thorough description of the input space or input data has been made. Abnormalities or characteristics about the data or input that need to be addressed have been identified.

Very nice job describing your dataset. Glad that you show some descriptive stats, show a sample of your data, go into a bit of detail in the features here and the distribution of the target variable. As this allows the reader to get an understanding of the structure of the data you are working with.

Maybe also look into computing the Kolmogorov-Smirnov test for goodness of fit.
(https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.kstest.html)

Need any data transformations for the features?

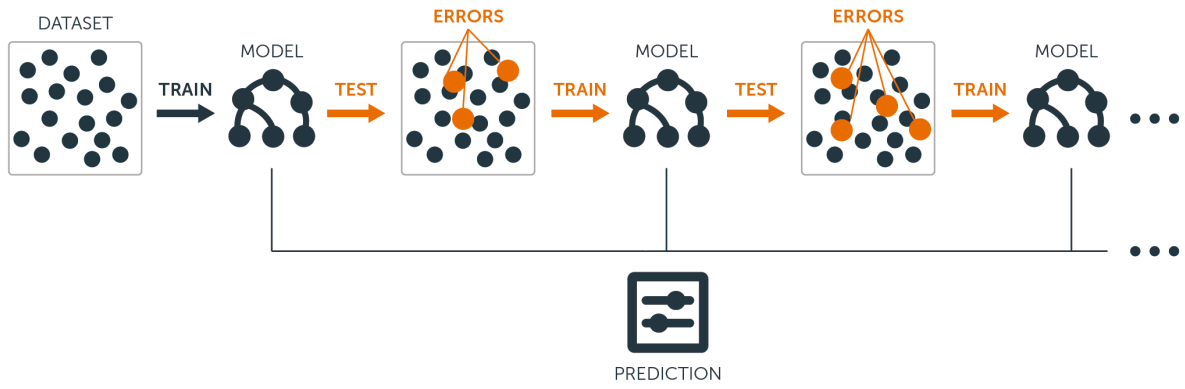
A visualization has been provided that summarizes or extracts a relevant characteristic or feature about the dataset or input data with thorough discussion. Visual cues are clearly defined.

All great visuals to show here! The distribution of target variable and box plots are always a good idea. Great use of seaborn!

You might also check out using some more advanced plotting libraries such as plot.ly: Modern Visualization for the Data Era. Where you can create really cool interactive visuals in jupyter notebooks and web apps!

Algorithms and techniques used in the project are thoroughly discussed and properly justified based on the characteristics of the problem.

You have some really nice analysis in this section, but here in this Algorithms and Techniques section, you should go into some more detail in how your algorithms of Logistic regression, Support vector Machine, and XGBoost are **trained**. For example, what is the kernel trick? Maximum margin for your SVM? etc... Don't be afraid to use mathematical formulas or visuals to explain these, if desired.



Student clearly defines a benchmark result or threshold for comparing performances of solutions obtained.

> "it is important to note that we are seeking to build a classifier that is able to find all of the rare events (i.e, customers who did churn) and therefore, the null accuracy is not relevant. In its place, we will use the percentage of the customers who did churn (26%) as the benchmark for this problem."

Good choice in a benchmark model. This is probably the best choice to use. Could also even look into running a very simple model (a simple KNN) to get a baseline score as well.

Methodology

All preprocessing steps have been clearly documented. Abnormalities or characteristics about the data or input that needed to be addressed have been corrected. If no data preprocessing is necessary, it has been clearly justified.

Excellent job documenting all your pre-processing steps. And cool idea to try out all the different Resampling techniques!

Nice job mentioning how you dealt with missing values. If you do want to get fancy, you could also run a supervised learning model to 'predict' the Nan value! There are many advanced techniques for imputing missing values including using machine learning and bayesian statistical approaches. This could be techniques as simple as using k-nearest neighbors to find the features that are most similar, and using the values those features have to fill in values that are missing or complex methods like those in the very popular AMELIA library. Regardless your imputation approach, you should be very cautious of the BIAS you are imputing into any model that uses these imputed values. Though imputing values is very common, and often leads to better predictive power in machine learning models, it can lead to over generalizations. In extremely advanced techniques in Data Science, this can even mean ethical implications. Machines can only 'learn' from the data they are provided. If you provide biased data (due to imputation, poor data collection, etc.), it should be no surprise, you will achieve results that are biased.

The process for which metrics, algorithms, and techniques were implemented with the given datasets or input data has been thoroughly documented. Complications that occurred during the coding process are discussed.

> "use the Scree plot to determine the optimal amount of principal components to keep."

Can you please elaborate a bit more on this comment. How much explained variance was kept? Your plot might be nice to see here.

Also what was the reason for doing this? Was this 'only' for visualization purposes? Or did you try using PCA features to train the model? As I am not quite sure why this is included in your model building pipeline (since there is no other discussion later on this).

Also for this section make sure you mention if any complications occurred during the coding process. Anything go wrong here? Was there any part of this process that was more difficult than the others? etc.. Stating any complications that occurred is always an important step in replicating your work.

The process of improving upon the algorithms and techniques used is clearly documented. Both the initial and final solutions are reported, along with intermediate solutions, if necessary.

Really like the Summary Results from All Modeling Runs table and your gridSearch and Resampling techniques, as these are excellent ways to improve your model. You have made it very clear in the parameter you tried and the results.

Maybe one other idea, since you have built multiple models, would be to 'ensemble' all of them. As we can typically 'squeeze out' a few more percentage points by doing so. Could look into using VotingClassifier

Results

The final model's qualities — such as parameters — are evaluated in detail. Some type of analysis is used to validate the robustness of the model's solution.

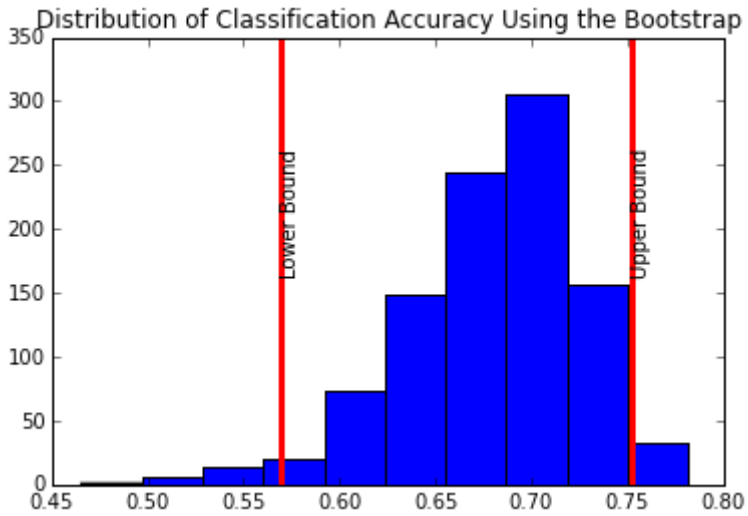
You have good analysis of your final models and excellent analysis to validate the robustness of the model's solution with your 3fold cross validation on all three models (maybe also show the variance of the results) and changing the number of training and testing points (could also run your model with multiple different random states).

Maybe one other idea would be to plot and 95% confidence interval to determine if the model is robust with bootstrapping. Here might be an example with the boston housing dataset.

```
from sklearn.utils import resample
import matplotlib.pyplot as plt

data = pd.read_csv('housing.csv')
values = data.values
# configure bootstrap
n_iterations = 1000
n_size = int(len(data) * 0.50)

# run bootstrap
stats = []
for i in range(n_iterations):
    # prepare train and test sets
    train = resample(values, n_samples=n_size)
    test = np.array([x for x in values if x.tolist() not in train.tolist()])
    # model
    model = DecisionTreeRegressor(random_state=100)
    model.fit(train[:,:-1], train[:,-1])
    score = performance_metric(test[:,-1], model.predict(test[:,:-1]))
    stats.append(score)

# confidence intervals
alpha = 0.95
p = ((1.0 - alpha) / 2.0) * 100
lower = max(0.0, np.percentile(stats, p))
```

```
p = (alpha + ((1.0 - alpha) / 2.0)) * 100
upper = min(1.0, np.percentile(stats, p))

# plot
plt.hist(stats)
plt.axvline(lower, color='red', lw=3)
plt.text(lower, n_iterations // 4, 'Lower Bound', rotation=90)
plt.axvline(upper, color='red', lw=3)
plt.text(upper, n_iterations // 4, 'Upper Bound', rotation=90)
plt.title('Distribution of Classification Accuracy Using the Bootstrap')
plt.show()

print('%.1f confidence interval %.1f%% and %.1f%%' % (alpha*100, lower*100, u
pper*100))
```

(https://machinelearningmastery.com/calculate-bootstrap-confidence-intervals-machine-learning-results-python/)



The final results are compared to the benchmark result or threshold with some type of statistical analysis. Justification is made as to whether the final model and solution is significant enough to have adequately solved the problem.

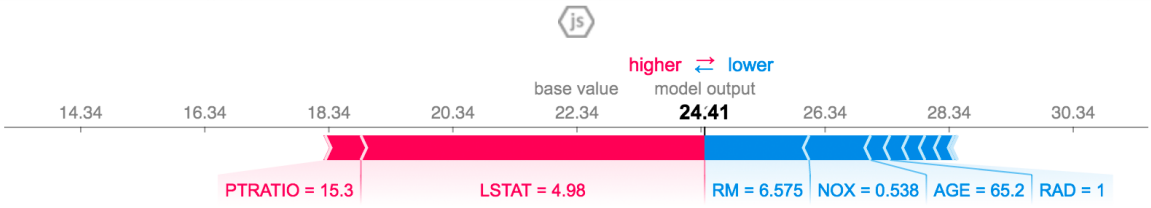
Conclusion

A visualization has been provided that emphasizes an important quality about the project with thorough discussion. Visual cues are clearly defined.

Make sure you include a **Free-Form Visualization** section. In this section, you will need to provide some form of visualization that emphasizes an important quality about the project. It is much more free-form, but should reasonably support a significant result or characteristic about the problem that you want to discuss.

Some ideas

- feature importance plot for your XgBoost
- confusion matrix
- Another really cool idea would be to check out the SHAP library. SHAP (SHapley Additive exPlanations) is a unified approach to explain the output of any machine learning model. SHAP connects game theory with local explanations, uniting several previous methods and representing the only possible consistent and locally accurate additive feature attribution method based on expectations (see the SHAP NIPS paper for details). This is where you can visualize your machine learning model's predictions with visuals such as



Student adequately summarizes the end-to-end problem solution and discusses one or two particular aspects of the project they found interesting or difficult.

Nice work discussing your final end-to-end problem solution as this reads quite well. I can definitely tell that you have spent a long time on this project as it really shows.

Discussion is made as to how one aspect of the implementation could be improved. Potential solutions resulting from these improvements are considered and compared/contrasted to the current solution.

> "Another improvement that I would propose for this project is to assess the performance of the model using other sampling methods including Synthetic Minority Oversampling TEchnique (SMOTE)"

My favorite over-sampling technique is SMOTE. Here is a good example.

Maybe even try and use a mixed input NN model. Check out this paper (https://arxiv.org/abs/1604.06737)

And here might be an idea of how this can be done in Keras (https://github.com/entron/entity-embedding-rossmann)

Quality

Project report follows a well-organized structure and would be readily understood by its intended audience. Each section is written in a clear, concise and specific manner. Few grammatical and spelling mistakes are present. All resources used to complete the project are cited and referenced.

Your writing is very clean and it is very easy to understand what you are saying. I personally thank you as this report is very easy to read :)

Code is formatted neatly with comments that effectively explain complex implementations. Output produces similar results and solutions as to those discussed in the project.

A few more comments in your actual code would be nice to see. Make sure you fully document what is going on in the relevant code cells. Remember that another engineer should be able to look at your code and know exactly what is going on.

You might also check out

- this post regarding Docstrings vs Comments.
- Google Style Python Docstrings
- This Best of the Best Practices" (BOBP) guide to developing in Python

RESUBMIT

DOWNLOAD PROJECT

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH

Rate this review