Machine Learning Engineer Nanodegree

Capstone Project – Starbucks app data

Matteo Felici
January 18th, 2020

I. Definition

Project Overview

“There’s an app for that” was the new motto from Apple in 2009. 10 years later, this statement is proven to be true, and every company has to create a proprietary app to sell his business. That’s why analysis on app usage is more crucial than ever to leverage the business: understand the customers’ behavior browsing the app, predict his needs and give the correct response.

Starbucks is one of the most well-known companies in the world: a coffeehouse chain with more than 30 thousand stores all over the world. It strives to give his customers always the best service and the best experience; as a side
feature, Starbucks offers his free app to make orders online, predict the waiting time and receive special offers.

This project aims at optimizing the customers' experience using the app through user's behavior analysis and one-to-one offers customization.

Problem Statement

Starbucks wants to find a way **to give to each customer the right in-app special offer**. There are 3 different kind of offers: *Buy One Get One* (BOGO), classic *Discount* or *Informational* (no real offer, it provides informations)
on a product.

Our goal is to analyze historical data about app usage and offers / orders made by the customer to develop an algorithm that associates each customer to the right offer type. We will develop 3 different models, one
for each type of offer: each model will evaluate the customer's propension towards that kind of offer returning a score. Given the 3 propensity scores, we will build an algorithm to make the final decision for each customer.

Metrics

Based on past data, we will compare the models prediction with the actual propension/not-propension of the customer through different performance metrics:

- **balanced accuracy:** it is the percentage of correctly classified records, corrected for unbalanced targets. The formula is

$$bal\_acc = \left(\frac{TruePositive}{Positive} + \frac{TrueNegative}{Negative} \right) \Big/ 2$$

  where *Positive* or *Negative* are the actual propensity of the customer, and the *True* label means that the model prediction is correct.
- **Precision/Recall:** this two measures focuses on the *Positive* labels:

$$precision = \frac{TruePositive}{PreictedPositive}, \qquad recall = \frac{TruePositive}{Positive}$$

  Basically, the first is the percentage of corrected positive labels on all the predicted positive labels, and answers to the question "*How much of the predicted positive have actually the propension?*". The second one is the percentage of corrected positive labels on all the actual positive records, and answers to the question "*How much of the positive records the model is recalling?*".
- **F1-score:** this measure is the armonic mean of the previous two:

$$F1 - score = \frac{2 * Recall * Precision}{Recall + Precision} = \frac{2TP}{2TP + FP + FN}$$

  A high F1-score ensures that the model is predicting correct positive labels, without leaving behind any actual inclined customer.

II. Analysis

(approx. 2-4 pages)

Data Exploration

For this project we have 3 available data sources.

Portfolio

It contains the list of all available offers to propose to the customer. Each offer can be a *discount*, a *BOGO (Buy One Get One)* or *Informational* (no real offer), and we've got the details about discount, reward and period of the offer. There are 10 available offers, for each one the features are:

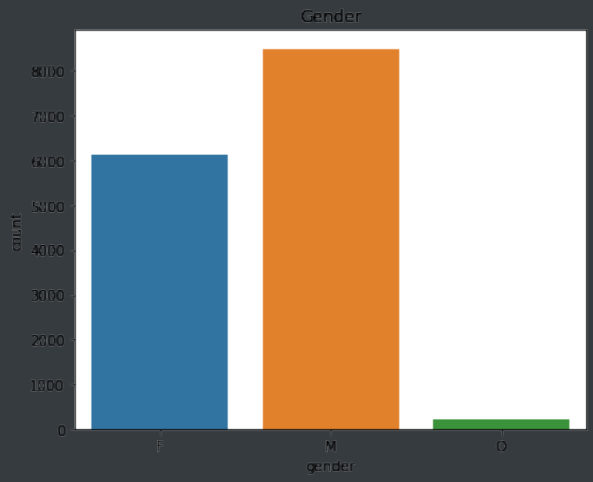
- *id* of the offer
- *offer_type*, it can be a *discount*, a *BOGO (Buy One Get One)* or *Informational* (no real offer)
- *channels* of the offer, a subset of web, email, mobile, social
- *difficulty*, the minimum amount the customer has to spend to activate the offer; for *Informational* it is always 0
- *reward* of the offer; for *Informational* it is always 0
- *duration* of the offer in days

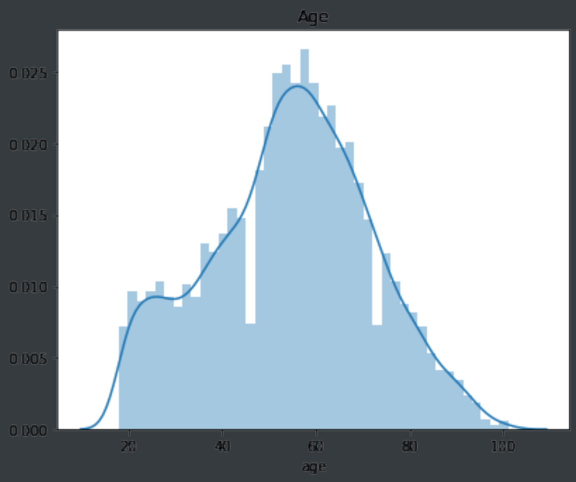
Profile

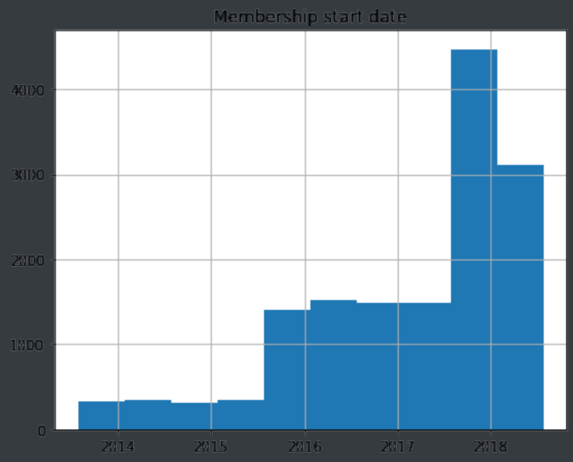
It contains the list of all customers that interacted with the app. There are 17,000 records. For each profile, the dataset contains some customer's information:

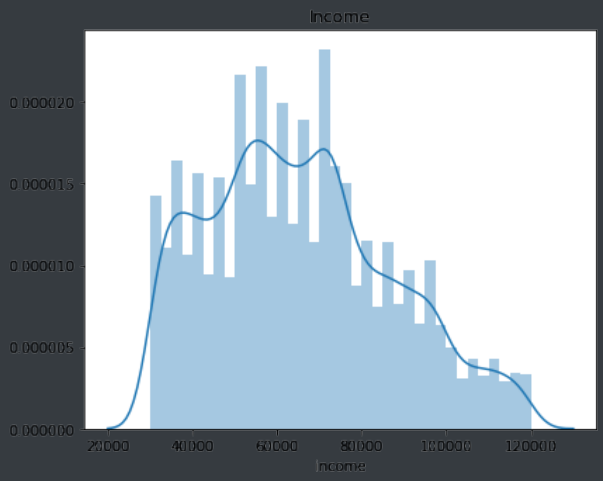
- *id* of the customer
- *gender*, it can be M, F or O
- *age*
- *income*
- *became_member_on*, date of when the customer became member of the app. The dates range from 2013-07-29 to 2018-07-26.

There are some missing values in *gender* feature: in the same rows, *income* is missing too, and *age* is always equal to 118. We can suppose thatn informations about these customer has been lost, so we can safely drop these records: there remain 14,825 records.









Taking a quick look at the distribution of the features on these customers, we can see that

- the customer's age is normally distributed, with a mean of 60 years old
- the income is left skewed, so there are few high-income customers
- there's a peak of subscriptions from the second half of 2017: maybe a big update of the app?

Transcript

This dataset has the list of all actions on the app relative to special offers, plus all the customers' transactions. There are 306,534 events. For each record, we've got:

- *person*, the id of the customer
- *event*, either **offer received**, **offer viewed**, **offer completed** or **transaction**
- *time* of when the event happened. It is measured in hours from a certain not specified **t0**. This value ranges from 0 to 714 (almost 30 days).
- *value*, a dictionary containing some additional data about the record. It can contains:
  - **offer_id** (the same of *Portfolio* dataset) for all events except transaction
  - **reward** of the offer for an "offer completed" event
  - **amount** spent for a transaction

For a better analysis, we split the *value* feature into 3 different columns.

Analysing the event types we can see that

- for each offer completed, there is a transaction with the same *time* value
- for each offer viewed there is an offer received
- there's no clear relationship between offer viewed and completed: an offer can be viewed but not completed, and vice-versa (a customer completes an offer without knowing)

New dataset – Customer journey

To understand a customer behavoiur, we have to recreate the **customer journey**: we create a new dataset, *journey*, starting from *transcript*, in this way:

- for each **offer received** we concatenate the relative **offer viewed**: we must be sure that the view is the nearest AFTER the reception, otherwise we could join the view with a subsequent reception of the same offer (a customer could receive the same offer multiple times)
- for each **offer viewed** we join the eventual **offer completed**, with the same logic. In this way, we drop the completion without view: since they are casual, they do not represent the customer's behavior
- for each **offer completed** we join the relative **transaction**
- finally, we concatenate all the remaining transactions that do not come from an offer completion

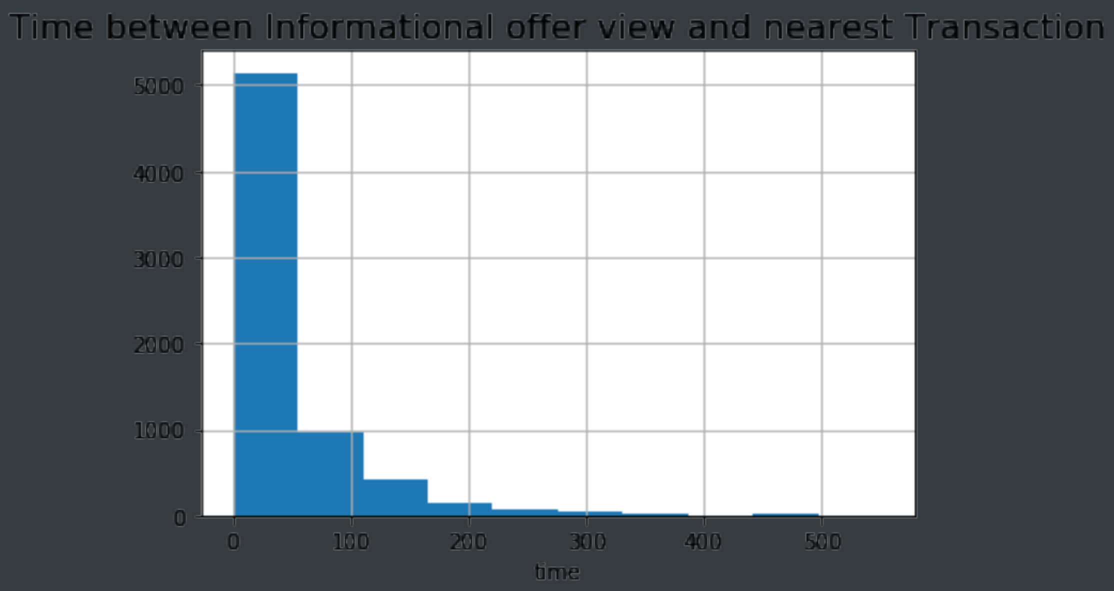
This data pipeline results in a dataframe with 188,234 records; we can now join data about the relative offer from **Portfolio** and the relative customer from **Profile**.

Feature Engineering

First of all, we build the 3 different targets. For **bogo** and **discount** we can put target = 1 (propension) when the offer viewed (of the relative type) ends with a completion. We take onlye the viewed, not received, because we're interested in the customer's behavior.

Since there's no real conversion (no *offer completed* record) for **Informational**, we develop a proxy. If we find a **transaction** right after an Informational offer view, we put that as propension. But how much hours is *right after*?

```
mean      43.230928
min        0.000000
25%        6.000000
50%       24.000000
75%       54.000000
max      552.000000
```



As we can see from the distribution of time between these two events, we can take the median value (24 hours) as *right after* value.

With these 3 targets, we have these Conversion percentages:

bogo	**50.47%**
discount	**65.73%**
informational	**38.82%**

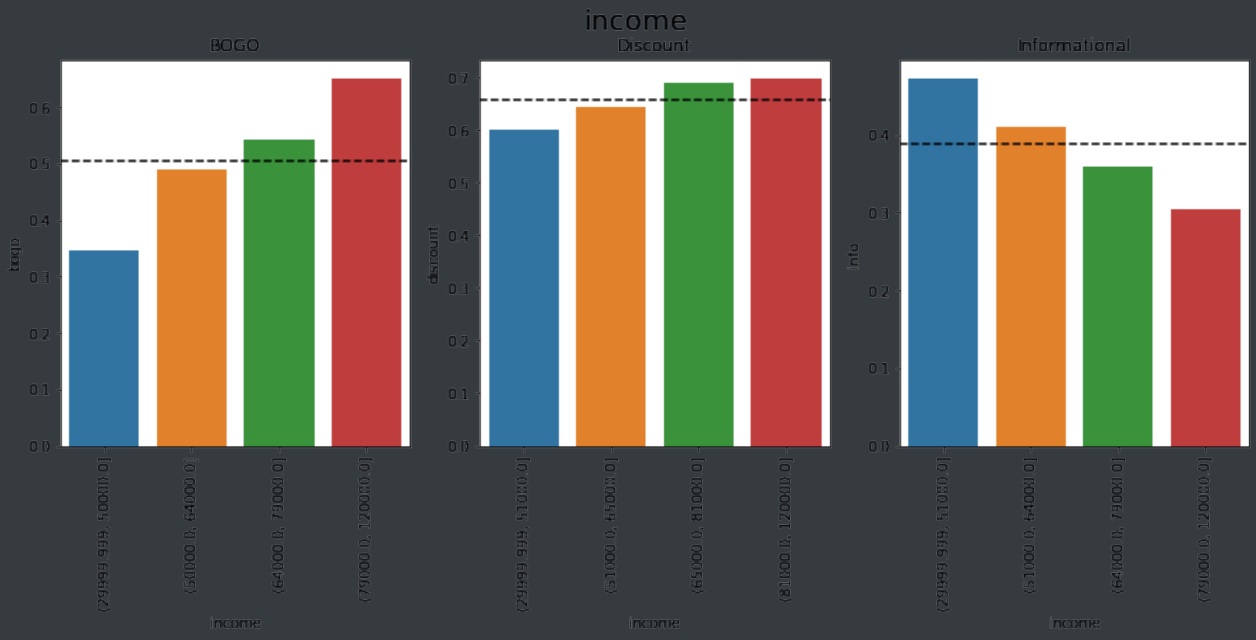
Starting from this dataset, we can enrich the data with new columns.

- *member_from* becomes number of months of membership
- *time* in hours is traduced in days; moreover, we count the days module 7 (days in the week), so the first 24 hours have label 2, the second 24 hours label 1 and so on
- we create a series of features based on past behavior, such as
  - number of transactions
  - average amount of transaction
  - number of completed / viewed offers
  - average reward received from offers
  - average time from reception to view, and from view to completion

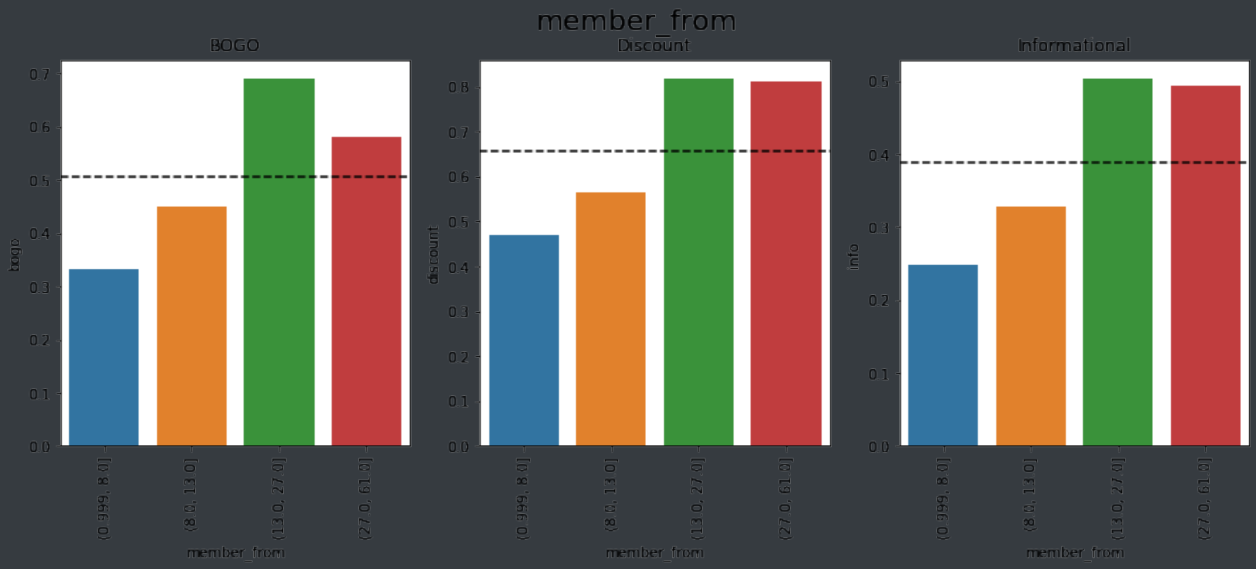
  All these features aim at understanding how much a customer is active on the app and is responsive to offers.

Exploratory Visualization

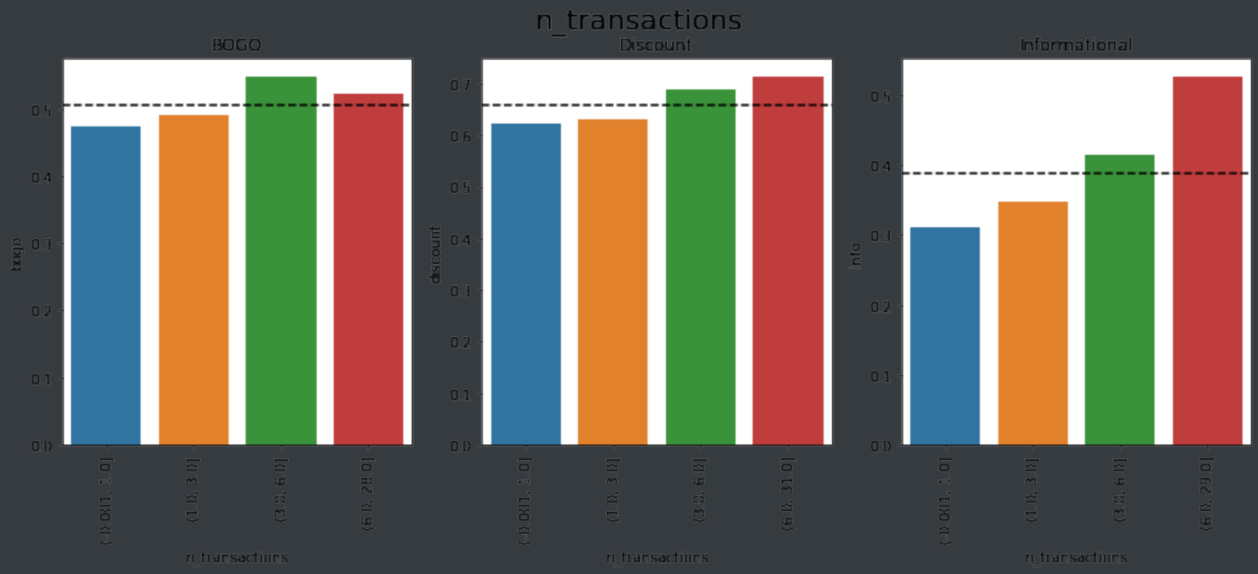
We cover now a subset of the input variables that has an interesting relation with the target(s). For each feature, we plot the average of the target for each category / bin; the black line is the target overall mean.



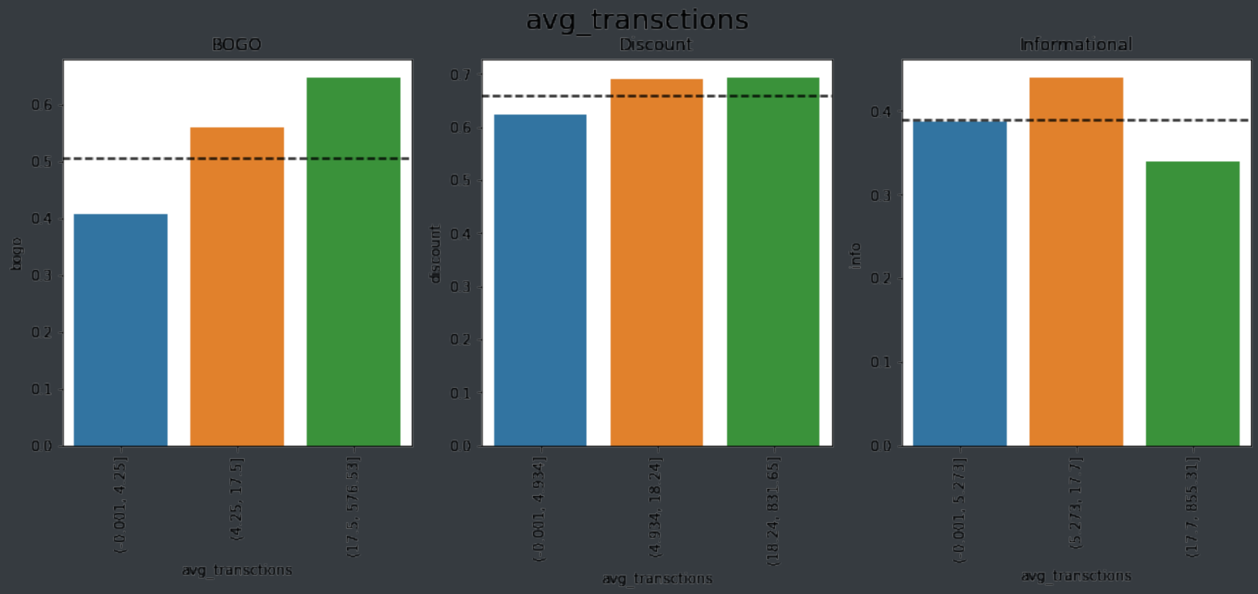
First of all, it seems like that higher incomes are more interested in special offers, especially **BOGO**. **Informational** offers instead have more effect on low income customers.



Being a senior member makes you more incline to complete each kind of offer. There's an interesting lower peak on the **BOGO** offering for the more senior customers.



This feature is relevant only for the **Informational** offers: it seems like more active customers are more interested in info on new products than discounts or special offers.



If we look at the average transaction, instead, we see that higher spending customers are more interested in special offers (especially **BOGO**) than information on products.

Algorithms and Techniques

To address the problem of understanding the best offer type for each customer we develop a series of Machine Learning models: taking as input past data, with the corresponding label of *propension* or *not propension*, the model will find relations between the input features and the event we want to predict (the offer completion) by building amathematical infrastructure re-usable on new customers.

We try 2 different type of algorithm for each target, then we choose the best one (comparing the performance measures listed in Secton 1 - *Metrics*) for each kind of offer. The algorithms are:

- **XGBoost:** this is an implementation of the *Gradient Boosting* algorithm, a widely used Machine

Learning model. This model develops a series of weak learners called *Decision Trees* in subsequent fashion: the first tree tries to predict the given target, while the second one aims at modeling the error of the first one and so on.

- **AWS Sagemaker's LinearLearner:** this is the Sagemaker's implementation of a classical linear model. It assigns a weight to each input feature to have an output score: if this score is higher of a certain *threshold*, the model predicts an event (offer completed), otherwise it predicts a non-event.

Benchmark

As benchmark measure, we decided to take the current **conversion rate** for each type of offer (described in this section's *Data Exploration*). After developing the models, we will compare these rates to the models' **precision scores**: since we will send the offer to all the customers that the model labels as "interested", the precision is exactly the expected conversion rate on that kind of offer.

III. Methodology

(approx. 3-5 pages)

Data Preprocessing

In this section, all of your preprocessing steps will need to be clearly documented, if any were necessary. From the previous section, any of the abnormalities or characteristics that you identified about the dataset will be addressed and corrected here. Questions to ask yourself when writing this section:

- *If the algorithms chosen require preprocessing steps like feature selection or feature transformations, have they been properly documented?*
- *Based on the **Data Exploration** section, if there were abnormalities or characteristics that needed to be addressed, have they been properly corrected?*
- *If no preprocessing is needed, has it been made clear why?*

Implementation

In this section, the process for which metrics, algorithms, and techniques that you implemented for the given data will need to be clearly documented. It should be abundantly clear how the implementation was carried out, and discussion should be made regarding any complications that occurred during this process. Questions to ask yourself when writing this section:

- *Is it made clear how the algorithms and techniques were implemented with the given datasets or input data?*
- *Were there any complications with the original metrics or techniques that required changing prior*

to acquiring a solution?
- *Was there any part of the coding process (e.g., writing complicated functions) that should be documented?*

Refinement

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant intermediate results as necessary. Questions to ask yourself when writing this section:

- *Has an initial solution been found and clearly reported?*
- *Is the process of improvement clearly documented, such as what techniques were used?*
- *Are intermediate and final solutions clearly reported as the process is improved?*

IV. Results

(approx. 2-3 pages)

Model Evaluation and Validation

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the model's solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section:

- *Is the final model reasonable and aligning with solution expectations? Are the final parameters of the model appropriate?*
- *Has the final model been tested with various inputs to evaluate whether the model generalizes well to unseen data?*
- *Is the model robust enough for the problem? Do small perturbations (changes) in training data or the input space greatly affect the results?*
- *Can results found from the model be trusted?*

Justification

In this section, your model's final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section:

- *Are the final results found stronger than the benchmark result reported earlier?*
- *Have you thoroughly analyzed and discussed the final solution?*

- *Is the final solution significant enough to have solved the problem?*

V. Conclusion

(approx. 1-2 pages)

Free-Form Visualization

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. Questions to ask yourself when writing this section:

- *Have you visualized a relevant or important quality about the problem, dataset, input data, or results?*
- *Is the visualization thoroughly analyzed and discussed?*
- *If a plot is provided, are the axes, title, and datum clearly defined?*

Reflection

In this section, you will summarize the entire end-to-end problem solution and discuss one or two particular aspects of the project you found interesting or difficult. You are expected to reflect on the project as a whole to show that you have a firm understanding of the entire process employed in your work. Questions to ask yourself when writing this section:

- *Have you thoroughly summarized the entire process you used for this project?*
- *Were there any interesting aspects of the project?*
- *Were there any difficult aspects of the project?*
- *Does the final model and solution fit your expectations for the problem, and should it be used in a general setting to solve these types of problems?*

Improvement

In this section, you will need to provide discussion as to how one aspect of the implementation you designed could be improved. As an example, consider ways your implementation can be made more general, and what would need to be modified. You do not need to make this improvement, but the potential solutions resulting from these changes are considered and compared/contrasted to your current solution. Questions to ask yourself when writing this section:

- *Are there further improvements that could be made on the algorithms or techniques you used in this project?*
- *Were there algorithms or techniques you researched that you did not know how to implement, but would consider using if you knew how?*
- *If you used your final solution as the new benchmark, do you think an even better solution exists?*

Before submitting, ask yourself. . .

- Does the project report you've written follow a well-organized structure similar to that of the project template?
- Is each section (particularly **Analysis** and **Methodology**) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?
- Would the intended audience of your project be able to understand your analysis, methods, and results?
- Have you properly proof-read your project report to assure there are minimal grammatical and spelling mistakes?
- Are all the resources used for this project correctly cited and referenced?
- Is the code that implements your solution easily readable and properly commented?
- Does the code execute without error and produce results similar to those reported?