# Machine Learning Engineer Nanodegree

## Capstone Proposal

June Yim

April 10th, 2019

## Domain Background

When graduate schools decide which applicants pass or fail, they consider many things including undergraduate GPA, GRE, TOEFL(only for foreign students), letters of recommendation, academic achievements and so on. Historically, applicants made efforts to do well in all these things or they prioritize them by their own standards and concentrate on the more important factors. Institutions which provide the admission consulting service for the applicants also have their own prioritization standards.

With applying Machine Learning algorithms to the admission data, we can make useful prediction model to help the applicants.

## Problem Statement

The problem is : “How much I can have confidence that I can get the acceptance from the graduate school with my records?” The records include GPA, GRE score, TOEFL score, and so on. (We well see in detail on the next section)

The confidence is expressed as a number in range [0, 1]. That is, we can think of the confidence as the probability of getting the acceptance.

So this problem is thoroughly quantifiable and measurable.

## Datasets and Inputs

I used Graduate Admission dataset from kaggle.

(https://www.kaggle.com/mohansacharya/graduate-admissions)

The dataset contains several parameters which are considered important during the application for Masters Programs.

The parameters included are : 1. GRE Scores ( out of 340 ) 2. TOEFL Scores ( out of 120 ) 3. University Rating ( out of 5 ) 4. Statement of Purpose and Letter of Recommendation Strength ( out of 5 ) 5. Undergraduate GPA ( out of 10 ) 6. Research Experience ( either 0 or 1 ) 7. Chance of Admit ( ranging from 0 to 1 )

Among these, ‘7.Chance of Admit’ is collected from the interview with the applicants.

I’ll set this - Chance of Admit - as the label and the remaining parameters(1~6) as inputs to make a prediction model.

Of course the results of applicaion will be a binary variable ‘1’(accepted) or ‘0’(rejected) but besides the difficulty of getting it, this results can change on the graduate schools’ inner or outer circumstances, for example, the number of applicants (competition) in that season. So, using the ‘Chance of Admit’ as a proxy for the binary result can be reasonable, I think.

Hence, the label in my model is in range [0, 1] instead of ‘0’ or ‘1’.

# Solution Statement

The solution to the problem is the confidence level that the applicants can expect for the graduate admission. The confidence level is a number between 0 and 1 (inclusive) so the applicants can think of this as the probability that he or she can get the admission with his or her records(GRE score, TOEFL score,...).

# Benchmark Model

There are many well-defined supervised learning algorithms. As benchmark models I will use them: linear regression, SWM, decision tree, random forest and so on.

# Evaluation Metrics

I will use RMSE(Root Mean Square Error) as the metric. As the prediction I want to do in this proposal is kind of a regression job, RMSE will be a good metric.

RMSE is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells us how concentrated the data is around the line of best fit.

For comparison, I will also use R-Squared at times as a supplementary metric. R-Squared is a statistical measure of fit that indicates how much variation of a dependent variable is explained by the independent variable(s) in a regression model.

# Project Design

To approach a solution I will explore the data to gain insight at first, and prepare & train the data to implement a model and fine-tune it to get the final model. I will explain the whole workflow step by step:

## 1. Get the data

- Download the data from the data source (kaggle)
- Convert the data to a format I can easily manipulate without changing the data itself (if necessary)
- Check the size and type of data
- Split the data to train and test set

## 2. Explore the data to gain insights

- Create a copy of the data for exploration
- Look into each attribute and its characteristics
- Visualize the data (plotting, etc.)
- Check correlations between attributes
- Identify promising transformations

## 3. Prepare the data for applying ML algorithms

- Data cleaning (take care of missing values, outliers, ...)
- Feature scaling

## 4. Try many different models and short-list the best ones

- Train many benchmark models including linear regression, SVM, Random Forests and so on

- Measure and compare their performance
- Find the most significant parameters
- Short-list the top two to three most promising models

## 5. Fine-tune the models to get the final one

- Fine-tune the hyperparameters (I will use cross-validation technique)
- Combine the best models to get the final model using ensemble methods
- Measure the final model’s performance with test set