ADMISSION CUTOFF FINDER

Final Deliverable

Engineering Admission Cutoff Learning System

Maryyam Niazi, Jan Ollers, Yaminah Qureshi, Nikou Kalbali
{niazim3, ollersjp, qureshiy, kalbaln}@mcmaster.ca

Supervisor: Dr. Frantisek Franek

Group 2

Contents

1 Abstract

Primarily worked on by: Nikou Kalbali

Admission Cutoff Finder is a tool that will help universities evaluate and increase the accuracy of their admission process. It uses applicant information from previous and current admission cycles to produce a grade cutoff threshold based on a determined ideal number of offerees to meet seat cap requirements. The thresholds determined by current admission processes can be inaccurate and negatively impact universities. If it is undervalued and the number of offerees are overshot, it can cause the university serious financial and space issues. If the determined threshold is overvalued, the number of offerees are undershot, and their can be significant missed opportunities for university funding via tuition fees and government support.

This system uses different machine learning algorithms, namely neural network and support vector machine approaches, on previous years of applicant data to produce models to help with predicting results. This is done by learning from trends and patterns to assist the university with current and future admissions processes.

2 System Requirements Analysis

Primarily worked on by: Maryyam Niazi, Yaminah Qureshi

2.1 Introduction

2.1.1 Background

A typical university admission process in Ontario takes two steps. The first step consists of determining a cutoff threshold to define a subset of all applicants that will be extended an offer if their grade average is not below the cutoff. Some of the offerees become acceptees, while others do not. The second step involves determining the number of applicants the university should be preparing to welcome to their institution and programs.

In this process, determining an ideal cutoff threshold such that the acceptees set is of a specific size with some small tolerance is difficult. If the threshold is undervalued, the number of offerees is overshot, typically leading to an acceptees set that is too large and costs the university financial and space issues from having to deal with too many students. If the threshold is overvalued, the number of offerees will be undershot, resulting in a set of acceptees that is too small and costing the university opportunities for revenue and funding from tuition fees and government support.

This project will use at least two different machine learning approaches trained on previous years of applicant data and their acceptance of offers to an Ontario university to tackle the problem of determining ideal cutoff thresholds for new applicant data pools based on the learned patterns.

2.1.2 Purpose

The purpose of this document is to define the software design of the outlined system.

2.1.3 Terminology

2.1.3.1 Offerees

The subset of all applicants who will get an offer of acceptance from the university given that their grade average is above the cutoff.

2.1.3.2 Acceptees

The subset of offerees who accept the offer received.

2.1.3.3 Grade Percentage Average (GPA)

The percentage average of the grades an applicant received as their midterm marks in their last year of high school.

2.1.3.4 Cutoff Threshold

The cutoff threshold represented by a grade percentage average that if met by an applicant will result in the applicant receiving an offer of acceptance.

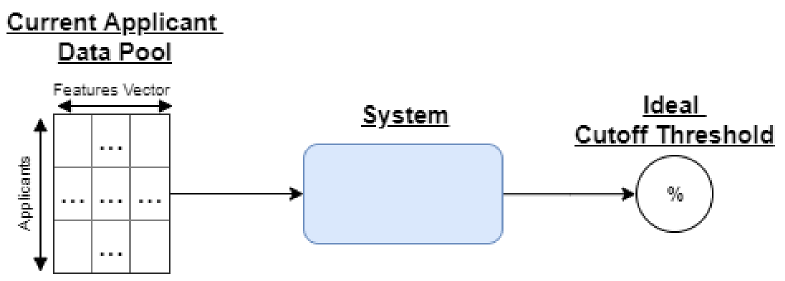


Figure 1: Conceptual generalized diagram of system

2.1.3.5 University Ranking

The ranking of preference given to the university submitted by an applicant.

2.1.3.6 Postal Code

The postal code of the location the applicant resides in.

2.1.3.7 Seat Cap

The maximum number of seats that a university can offer for a program for a particular year.

2.1.4 Statement of Goals

The system will be used for determining cutoff thresholds for two programs at an Ontario university.

By taking in applicant information from the current admission cycle and using information from previous admission cycles, the cutoff thresholds will be estimated to obtain an ideal number of offerees, such that the predicted number of acceptees should be as close to the target seat cap as possible.

2.2 System Overview

2.2.1 Use

The system will be used for determining cutoff thresholds for two programs at an Ontario university.

By taking in applicant information from the current admission cycle and using information from previous admission cycles, the cutoff thresholds will be estimated to obtain an ideal number of offerees.

2.2.2 Users

The primary intended users of the system are the Admissions Committees for Ontario universities. The system may be used as an analysis tool to help them evaluate and/or improve their current admissions process. Further, if we can expand the system and make it applicable to other university program admissions, the admissions committees of these programs will serve as secondary users.

2.2.3 Environment

It is intended that the system will be runnable by MAC, Linux/Unix and Windows operating systems as a command-line program. Users will be required to upload the data they would like to input to the system in a directory structure/format as specified and the system will use the data to output a result directly to the terminal/command prompt. Additional files may be created with further information about the result and be stored in a directory structure/format.

2.3 Context Diagram

2.4 Functional Requirements

2.4.1 System Data Requirements

The system should be built around 8 years of data which will include applicant pool data (e.g. postal code, university rankings, grade percentage averages, offer status, acceptance status, etc.) and the seat caps for each year.

2.4.2 Input Requirements

The system should accept applicant pool data for one year and currently available seating cap for the same year. The applicant pool data includes information about the student postal code, university preference rankings and midterm grade percentage averages.

2.4.3 Output Requirements

The system should output a percentage representing the cutoff threshold that applicants must meet to receive an acceptance offer.

- The system will produce a percentage of up to two decimal places that will not be rounded up.
- The number produced must be such that the set of projected acceptees is of a specific size, within a degree of tolerance. This size is influenced by the number of seats that the University wishes to offer for the program for the year. The tolerance is such that the percentage produced to the 0.01 degree should minimize the difference between the number of acceptees and the number of available seats for the program, such that the number of acceptees is not zero [1].

2.5 Non-functional Requirements

2.5.1 Confidentiality

The data that will be used to implement and test the system will include sensitive information about students who have applied to two particular programs at a specific university. It is important to make sure this information remains confidential throughout the development of the system. Some measures that will be taken to ensure this include: completely anonymizing the information before use, restricting the system from outputting or storing any data that could compromise student confidentiality, and restricting users of the program from accessing data they have not provided themselves. [2]

2.5.2 Reliability

The system should produce an accurate and realistic result for all data that meets the required input data structure.

2.5.3 Data Integrity

The data taken in as input to the system should not be modified before it is used by the system.

2.5.4 Reusability

Ability to use the system over multiple years of data, as long as admission processes and policies do not significantly change.

2.6 System Constraints

- The scope of the system is limited to two particular programs at a specific university as the data that has been provided on which the system will be trained is for these particular programs. For this reason, the system is limited to producing results for that university's admissions. However, in the future, expanding this project to more universities and programs is a possibility.

[1] Universities are not allowed to discriminate based on a grade (i.e. all applicants must be offered an acceptance if they meet the cutoff threshold)

[2] Questions regarding confidentiality of the data may be directed to Dr. Frantisek Franek

- Although using more data to develop the system may result in increased accuracy, only data from the previous 8 years of applicants are available as universities are required to maintain applicant confidentiality. For this reason, the system is constrained to learning from 8 years of previous applicant information.

3 Software Design/Architecture

Primarily worked on by: Maryyam Niazi, Yaminah Qureshi

3.1 Introduction

3.1.1 Background

A typical university admission process in Ontario takes two steps. The first step consists of determining a cutoff threshold to define a subset of all applicants that will be extended an offer if their grade average is not below the cutoff. Some of the offerees become acceptees, while others do not. The second step involves determining the number of applicants the university should be preparing to welcome to their institution and programs.

In this process, determining an ideal cutoff threshold such that the acceptees set is of a specific size with some small tolerance is difficult. If the threshold is undervalued, the number of offerees is overshot, typically leading to an acceptees set that is too large and costs the university financial and space issues from having to deal with too many students. If the threshold is overvalued, the number of offerees will be undershot, resulting in a set of acceptees that is too small and costing the university opportunities for revenue and funding from tuition fees and government support.

This project will use at least two different machine learning approaches trained on previous years of applicant data and their acceptance of offers to an Ontario university to tackle the problem of determining ideal cutoff thresholds for new applicant data pools based on the learned patterns.

3.1.2 Purpose

The purpose of this document is to define the software design of the outlined system.

3.1.3 Statement of Goals

The system will be used for determining cutoff thresholds for two programs at an Ontario university.

By taking in applicant information from the current admission cycle and using information from previous admission cycles, the cutoff thresholds will be estimated to obtain an ideal number of offerees, such that the predicted number of acceptees should be as close to the target seat cap as possible.

3.1.4 Users

The primary intended users of the system are the admissions committees for Ontario universities. The system may be used as an analysis tool to help them evaluate and/or improve their current admissions process. Furthermore, if we can expand the system and make it applicable to other university program admissions, the admissions committees of these programs will serve as secondary users.

3.2 Functional Description

3.2.1 Data

The system will be developed using applicant pool data from 8 years of admission cycles for two programs at an Ontario university. We describe some nuances related to the data in the following section.

3.2.1.1 Extracting Offeree Data

As our system will be predicting whether or not a student accepted an offer, we will only consider students that met the cutoff average for their application year for the University programs in question. This will be done by determining the GPA cutoff threshold for each admissions year and extracting the data for students that met this threshold.

3.2.1.2 Applicant Features

Our data contains multiple attributes about applicants in an admissions cycle, however, we only consider the following attributes of each student and will refer to these as **features**:

1. **Applicant GPA** - The system will consider GPA to two decimal places. For the focus university and program, the average will be computed using the following high school course set:
   - Math 1 - Functions
   - Math 2 - Calculus
   - English
   - Chemistry
   - Physics
   - Next best course
2. **Location** - The approximate distance of the applicant's postal code to the Ontario university in question.
3. **University Preference Ranking** - The preference ranking the applicant has given to the Ontario university in question, where 1 indicates it is their top choice.
4. **Secondary School** - The secondary school the applicant attended.
5. **Gender** - The gender of the applicant.
6. **Offer Decision** - Whether the applicant accepted an offer to the Ontario university in question, where 0 indicates they did not accept the offer and 1 indicates they did.

3.2.1.3 Partitioning the Data

To test the accuracy of our system we will need to use data that has not been used to train the system. The 8 years of data will be split 80-20 as training and testing data respectively. Only the training data will be used to fine-tune the system.

3.2.1.4 GPA Cutoff Threshold

The available data does not explicitly contain information about the final GPA cutoff thresholds that were determined for each admissions cycle. Instead, this will be determined by extracting the lowest applicant GPA amongst all applicants that received an offer per admissions cycle.

3.2.1.5 Target Seat Caps

In addition to the applicant data described above, the target enrollment the Ontario university set for each of the programs in question for each admissions cycle will also be available. This will be used in the final step in predicting a GPA cutoff threshold.

3.2.2 Approach - (Algorithm Viewpoint)

The goal of our system is to determine the ideal cutoff threshold such that the difference between the number of acceptees and the target seat cap for an admissions cycle is minimized. The proposed architecture to accomplish this will use the data regarding the features of each applicant in the current admissions cycle to predict whether the student will accept an offer to the programs in question if it were to be extended. Once such a prediction has been made for each applicant in the admissions cycle, the applicants will be sorted in descending order of their GPA. We will then iterate through each of the applicants until we have passed through a number of applicants that are predicted to accept the potential offer equal to the target seat cap for the current admissions cycle. The GPA of the applicant at which we reach this number will be selected as the GPA cutoff. In the case that several applicants have a GPA equal to this GPA cutoff predicted by our system, we will choose either this GPA cutoff or a GPA cutoff 0.01 percent above it by determining which option will result in a number of applicants predicted to accept a potential offer closer to the seat cap for the current admissions cycle.

To accomplish this purpose, a model that predicts whether an applicant will accept a potential offer must first be developed. Since the data we're working with is considered **labelled** (i.e. for each feature vector, it is known whether the associated student accepted/declined the extended offer), supervised machine learning approaches would be ideal. Two supervised machine learning methods will be used to develop these models: deep learning with neural networks and support vector machines.

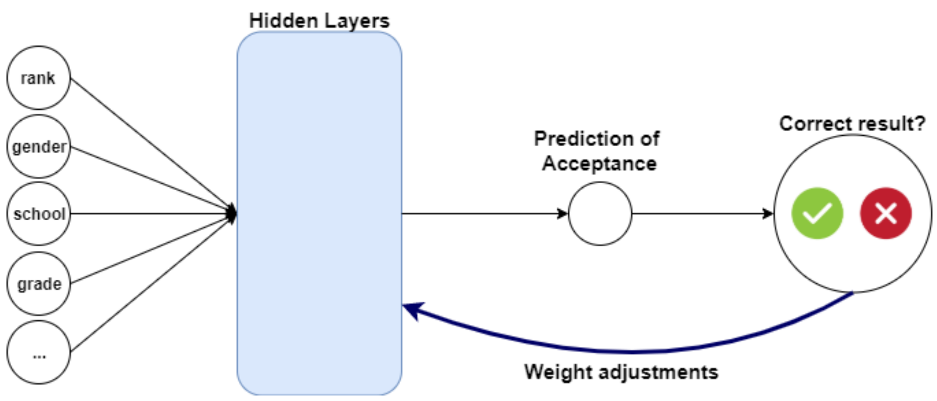


Figure 2: A visual of the architecture used in the deep learning approach

3.2.2.1 Deep Learning Approach

Deep learning refers to the application of multiple non-linear mathematical transformations to some input data in an attempt to extract information about the structure and patterns underlying the data. These transformations make up a **neural network**. More specifically, a neural network is comprised of "neurons" that take some input, compute a weighted sum of the input data, and apply a non-linear function known as an **activation function** to the weighted sum. This result is outputted by the neuron. In a neural network, neurons are typically organized into multiple layers and each neuron in a layer is connected to neurons in other layers such that the output of a neuron serves as the input to other neurons. To create a neural network tuned to a specific application, the values of the weights and parameters in the transformations made by the neural network must be determined. This is done by using some ground truth data which knows the desired output of the neural network for a given input to train the network. To determine how far the current neural network is from outputting optimal accuracy, an error function is developed. Weights are modified to minimize the value of the error function by computing the gradient of the error function with respect to the weights in the neural network through the use of a technique termed **backward propagation**.

The neural network we propose will accept as input the features listed above and will output either 0 and 1 predicting whether the student will accept an offer if extended; 0 implies the student will not accept the offer and 1 implies that they will (Figure 2). We will start initially with randomized values for the weights and parameters in the neural network and an arbitrary number of layers, and use the commonly used sum of squares of absolute errors as our error function: $|desired - actual|^2$. We will use backward propagation to continually improve the accuracy of the neural network and will also experiment with the number of layers and structure of layers used.

3.2.2.2 SVM Approach

As in the machine learning approach, the purpose of the support vector machine approach is to essentially classify the data into one of two classes: a class where the students will accept the offer, and the other where they won't. **Support vector machines (SVMs)** are a supervised machine learning approach to determine an optimal hyperplane for data classification when the data is visualized in multiple dimensions. For our application, a kernelized SVM may be used to obtain increased accuracy, wherein functions called **kernel tricks** will use dot products to map data points into higher dimensions to transform non-linear data into linear space and improve classification accuracy.

Since speed is not a requirement for this project, the potential drawback that comes with using SVMs leading to larger processing times is superficial[3]. Be that as it may, as mentioned earlier, kernel functions, as well as tuned hyperparameters, will be incorporated to increase speed.

One of the reasons why the SVM is considered a viable approach for this project is because SVMs are effective in high dimension spaces[3] and in this classification process, multiple dimensions result from the multiple features that must be considered for each student.

The steps that will be taken to implement this approach can be broken down into 3 main steps as listed below:

1. Start with data that the model will be trained using (as outlined in 3.2.1.1).
    - As with the deep learning approach, data will be input as feature vectors.

[3]See source: https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589

- Within the implementation, the decision boundary will be drawn arbitrarily at first, and will continue to improve once more data is passed through the learning model.

2. "Transform" data to higher dimensions to optimize accuracy of decision boundary.
    - Data will be transformed into higher dimensional spaces to allow for optimal accuracy for the determined hyperplane, using the aforementioned "kernel trick" to reduce computational costs as much as possible.
    - Part of the development process includes manipulating kernel types to see how the model can best be improved in terms of accuracy. Some popular kernel types include polynomial kernels, RBF kernels, sigmoid kernels, etc.. Tuning will likely be required for the kernel parameters as well.
3. Find the support vector classifier that classifies higher dimensional data into 2 groups by continuously validating and optimizing the found hyperplanes.

Due to the assumption that the data is likely well-behaved enough to determine a trend, not too many outliers and extreme examples are expected, however, in the case of such outliers, a soft-margin approach will allow the model some misclassification for the sake of optimizing classification in the long run. Part of the task is to research and tweak parameters to optimize bias/variance tradeoff for our application. Through training, the model should be able to draw a decision boundary with some margin from the potential extreme points in either classified group to allow for optimal classification by taking into account potential outliers in the data.

3.2.3 Experimental Analysis

An observation that helped gear model optimization was the fact that roughly 23% of offers to the target university were declined. This was a helpful observation because it meant there was an imbalance of data samples for the two classes of acceptees and declinees that needed to be separated and it cautioned of a strong bias towards the majority class of acceptees. Since SVMs tend to not be effective when data classes are imbalanced, the SVM approach was especially influenced. [4]

Namely, it helped tune the SVM model by

- guiding to the selection of a linear kernel as opposed to the more commonly used polynomial and radial basis function kernels evident due to consistently higher prediction scores by up to 5% of a difference
- leading to an understanding for why class weights could not be set to "balanced" since it would lead to highly inaccurate predictions by favouring the many more data points in the acceptee class
- increasing the regularization parameter (also called **C** or lambda) to reduce the inaccuracy caused by the skewed class balance; a large value for this parameter specified to the model that a softer margin for the hyperplane should be used, decreasing chances of overfitting to the training data (which is easy to do when there are few minority case samples) [5]

On the other hand, in order to reduce the negative impact the imbalanced classes had on the neural network's prediction accuracy, data normalization brought up the prediction score from consistently around the low 80% mark to a vastly improved 96%. The normalization scaled each feature such that it was between the values 0 and 1 [6] , which transformed the data so that the mean of the data is close to zero, thereby ensuring that the "magnitude of the values that a feature assumes is more or less the same". This improves the neural network's learning process since we are not aware in advance which features are more important than others. [7]

While both approaches required different analyses, research, and optimization techniques to eventually increase and obtain the prediction scores of up to a max of 96%, this result is constrained by the dataset size. In order to start model training for both approaches, the data first needed to be "clean" to ignore any records that were missing crucial features, which caused roughly a 13% reduction in data record count. Furthermore, from within those records, only the students that were offered an acceptance were relevant, resulting in a significantly lower data count. Overall, the mediocre results were indicative of the fact that there is room for improvement to model training by accumulating more data to train on.

[4]See source: https://machinelearningmastery.com/cost-sensitive-svm-for-imbalanced-classification/

[5]See source: https://towardsdatascience.com/dealing-with-imbalanced-classes-in-machine-learning-d43d6fa19d2

[6]See source: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

[7]See source: https://towardsdatascience.com/why-data-should-be-normalized-before-training-a-neural-network-c626b7f66c7d

3.2.4 Limitations

Some limitations of the proposed system are outlined as follows.

3.2.4.1 Size of Dataset

One major limitation that must be considered is the relatively small size of our dataset. Our training data contains data for approximately 6 admissions cycles and for each of these cycles only data for students that received an offer are pertinent. The concern with this is that of overfitting. When training data is limited, the resulting model may not be as robust or general and instead correspond too closely to the data it was trained with. This can be reconciled however by taking into consideration that in general we expect future admissions cycles to be quite similar to the ones we are training with. Our application is one for which we do not expect many edge cases or noise.

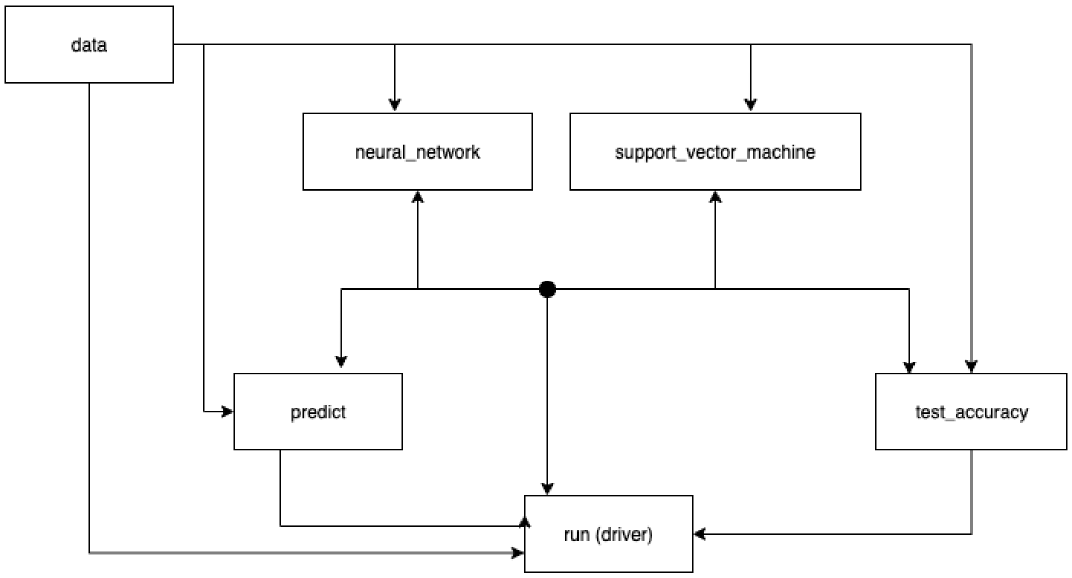
3.2.4.2 Computation Efficiency

In general, when talking about machine learning approaches a major limitation is the amount of time and computation power needed. Since our dataset is smaller and constant in size this is a secondary concern. The focus will be on the accuracy of the system rather than its speed or efficiency.

3.2.5 Software Architecture - (Structure,Interaction Viewpoint)

The following describes a primitive outline of the software architecture we will follow:

File name	Uses	Description
run	data, neural_network, support_vector_machine predict, test_accuracy	Driver: Predicts the cutoff average for a specified year of data and target cap and displays the accuracies of the models.
data	N/A	Parses, cleans and extracts relevant data from input data. Splits the data into train and test data and stores it in a file.
neural_network	data	Defines the neural network that takes as input data about an applicant and predicts whether the applicant will accept a potential offer. Trains the model using the train data, determining weights and saves the trained model.
support_vector_machine	data	Defines the support vector machine that takes as input data about an applicant and predicts whether the applicant will accept a potential offer. Trains the model using the train data, determining weights and saves the trained model.
predict	neural_network, support_vector_machine, data	Uses the test data and trained models to predict whether offerees would accept offers and the cutoff threshold.
test_accuracy	neural_network, support_vector_machine, data	Compares the predictions made from the test data with the ground truth to determine the models' accuracies.

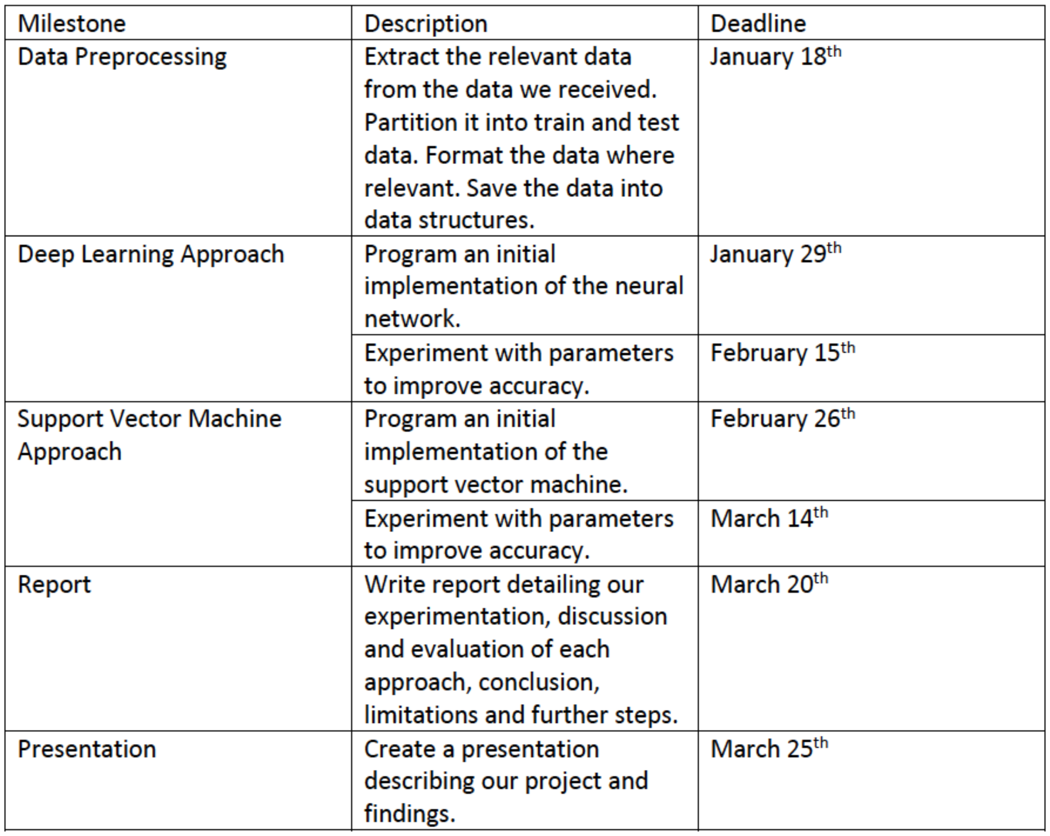

data
neural_network
support_vector_machine
predict
test_accuracy
run (driver)

3.3 User Interface - (Interface Viewpoint)

The system will involve little user interaction and as such will be delivered in the form of a command-line application. It is intended that the system will be run on MAC, Linux/Unix and Windows operating systems. Users will be required to upload the data they would like to input to the system in a specified input directory and format. The application will then use the data and run it through the system to output a result directly to the terminal/command prompt describing the GPA cutoff threshold predicted by the system, how many offers to be sent out, and how many offerees are predicted to accept the offer. Additional files will be created with further information about the results and be stored in a specified output directory. More details and specifics related to user interaction with the system can be found in the README referenced in section 4.

3.4 Milestones

The following describes a general outline of the milestones our project is divided into:



Milestone	Description	Deadline
Data Preprocessing	Extract the relevant data from the data we received. Partition it into train and test data. Format the data where relevant. Save the data into data structures.	January 18th
Deep Learning Approach	Program an initial implementation of the neural network.	January 29th
	Experiment with parameters to improve accuracy.	February 15th
Support Vector Machine Approach	Program an initial implementation of the support vector machine.	February 26th
	Experiment with parameters to improve accuracy.	March 14th
Report	Write report detailing our experimentation, discussion and evaluation of each approach, conclusion, limitations and further steps.	March 20th
Presentation	Create a presentation describing our project and findings.	March 25th

Note: as majority of our project is experimentation in an attempt to improve the accuracy of our models, it does not very closely follow the traditional milestones and architecture of software projects.

4 Code Guide - (Resource Viewpoint)

Primarily worked on by: Nikou Kalbali, Yaminah Qureshi

The source control repository can be accessed here: `https://github.com/Tallwhitebro/CapstoneProject/`.

The system can be compiled and run on MAC, Linux/Unix, and Windows systems as a command line program. The implementation is written in Python, version 3.7.1.

Required Modules: The following open source programs need to be installed on your computer using `pip` or `anaconda`:

```
pip3 install -U scikit-learn scipy matplotlib
pip3 install pandas
```

```
pip3 install numpy
pip3 install tensorflow==2.0
pip3 install keras
```

README: Once the required modules have been successfully installed, execute the `src/run.py` file.

Implemented Modules: Below is a list of all of the Python modules included in the **Code** folder, along with a brief description of each.

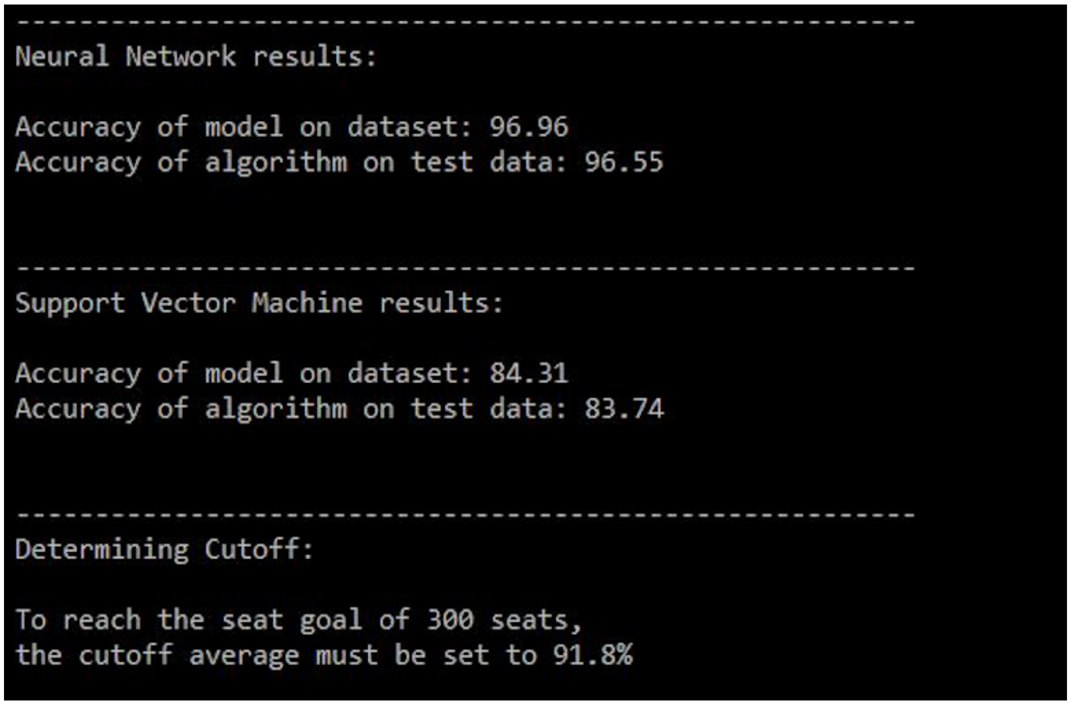
1. **postalCodeChecker.py:** Uses student postal codes to find their approximate distance to the university in question.
2. **data.py:** Parses and extracts relevant input data for storing into a data structure.
   - **prepare_data()** Loads the cleaned data for all 8 years of students that received an offer of admission and does some additional parsing. Splits the data into 4 groups: X_train (applicant features for students in the train data), y_train (whether the corresponding student in the training data accepted an offer), X_test (applicant features for students in the test data), y_test (whether the corresponding student in the test data accepted an offer). The data from all 8 years are scrambled and split into 2 groups where 80% of the students in the data pool make up the training data and the remaining 20% is used to make up the test data. Note: rerunning this method will randomize and split the data amongst the 4 groups again. For this reason it should not be run, rather, the retrieve method outlined below should be used to access the saved data.
   - **retrieve_data()** Retrieves the previously created and saved X_train, y_train, X_test and y_test data and loads it for use.
3. **neural_network.py:**
   - **create_model()** The Keras library is used here to create a simple neural network model. The model is then trained using the train data. This helps determine appropriate parameters for the model. We save this model in a file.
   - **retrieve_model()** Retrieves the previously created and saved model and loads it for use.
4. **support_vector_machine.py:**
   - **create_model()** The sklearn library is used to create a simple support vector machine. The model is then trained using the train data. This helps determine appropriate parameters for the model. We save this model in a file.
   - **retrieve_model()** Retrieves the previously created and saved model and loads it for use.
5. **test_accuracy.py:**
   - **accuracy_NN()** Determines and displays the accuracy of the neural network model and algorithm. This is done for the model by predicting whether a student will accept an offer for the students represented by the X_train data and comparing the prediction to the y_train data and for the algorithm by predicting whether a student will accept an offer for the students represented by the X_test data and comparing the prediction to the y_test data using the saved neural network model.
   - **accuracy_SVM()** Determines and displays the accuracy of the support vector machine model and algorithm. This is done for the model by predicting whether a student will accept an offer for the students represented by the X_train data and comparing the prediction to the y_train data and for the algorithm by predicting whether a student will accept an offer for the students represented by the X_test data and comparing the prediction to the y_test data using the saved support vector machine model.
6. **predict.py:**
   - **predict(model, file_to_path)** Using the given model, predicts whether students represented by the X_test data will accept an offer if extended and saves the prediction to the file in the specified path.
   - **predict_NN()** Makes the prediction for the students represented by X_test using the saved neural network model.

- **predict_SVM()** Makes the prediction for the students represented by X_test using the saved support vector machine model.
- **allocate_students(sorted_students, max_capacity, model)** Determines the lowest average for which an amount of students equal to max_capacity will accept an offer if extended (where students are represented in the sorted_students data by their applicant features and their acceptance of an offer is determined by making a prediction using the provided model).
- **calculate_cutoff(data_from_path, target_seat_cap, model)** Retrieves and parses data about students that applied to the university for a given year, where the location of the data is specified by the file path (data_from_path). This data is used to determine the cutoff average that must be output to have an amount of students predicted to accept an extended offer equal to the target_seat_cap. The passed model is used to make the prediction as to whether a student in the dataset will accept an offer if extended.

7. **run.py** Runs the code to determine what the cutoff GPA must be such that a number of students equal to target seat cap are predicted to accept an offer of admission to the university. The predictions are made by first using the neural network model and then the support vector machine model, the results of both are displayed. The accuracy of the neural network and the support vector machine are also displayed. The target seat cap as well as the year of students for which the code should be run on can be specified by changing the values in the input.yaml file.

5 Conclusions

Below are sample results for predicting an ideal cutoff threshold given a target seat cap of 300. The neural network model and support vector machine model were trained with the same data but the neural network model consistently made more accurate predictions. We determine the final cutoff average using the more accurate model, in this case, the neural network model which has a prediction accuracy of 96.96%. The final cutoff average was predicted to be 91.8%.



```
-------------------------------------------------------------
Neural Network results:

Accuracy of model on dataset: 96.96
Accuracy of algorithm on test data: 96.55


-------------------------------------------------------------
Support Vector Machine results:

Accuracy of model on dataset: 84.31
Accuracy of algorithm on test data: 83.74


-------------------------------------------------------------
Determining Cutoff:

To reach the seat goal of 300 seats,
the cutoff average must be set to 91.8%
```

Due to the nature of this project, there is still room for improvement be it in terms of optimizing the approaches implemented for our system, or by way of researching and implementing very different machine learning approaches. One major improvement can be made by increasing the size of the dataset by collecting more applicant data throughout the years to come. Machine learning is a broad field with many opportunities applicable to this problem yet to be explored.