STUDENT BIO

Name : Indraneil Paul

Education Level : International Institute of Information Technology - Hyderabad, Senior Year
Course/Major : Computer Science and Engineering
Degree Program : B.Tech and MS by Research in Data Science (Integrated Dual Degree)
Mail : indraneil.paul@research.iiit.ac.in or androneil54@gmail.com

Prediction of Energy Consumption of Electric Vehicles

3th April 2017

OVERVIEW

This project aims to develop a machine learning approach to predict the energy consumption of an electric vehicle given information about the intended route and associated terrain information such as altitude. This project will also however, have to find an appropriate dataset from which the models can be learnt and settle nuances such as finding an appropriate sampling rate for the time series data.

Additionally, the possibility of not finding the desired type or the adequate amount of data will also have to be accommodated, in which case a host of simulation approaches might have to be looked into. Eventually, this project aims to allow users to predict energy consumption after having proposed routes and hence may also be used to estimate operating ranges, thus helping alleviate range-anxiety.

GOALS

1. Curate a dataset that contains the required information such as GPS coordinates, altitude, etc.
2. In case of inadequate data, generate some data of our own using simulation techniques.
3. Explore a host of machine learning approaches and compare them to see which of them works best on the time series data at various sampling rates.

ABOUT ME

I am a fourth year Computer Science student at International Institute of Information Technology, Hyderabad. I am enrolled in an integrated dual degree programme of Bachelors in Computer Science and Masters in Data Science.

I have been an avid linux user since my school days and am proficient in several languages like C++, C, Python, Js, etc. I am also comfortable using several machine learning frameworks and toolkits such as TensorFlow, Keras, Lua, Scikit, Pandas, NumPy, CUDA, etc. which I have used in my coursework as well as personal projects. I have previously developed an asynchronous multi-agent messaging and coordination service in Python and ZeroMQ while interning for the Hyderabad start-up Robustest.

I have also taken several relevant data science courses in my college such as Statistical Methods for Artificial Intelligence, Information Retrieval and Extraction, Computer Vision, Machine Learning, NLP Applications.

As a regular staple of my coursework, I have implemented several machine learning routines. In my Statistical Methods for Artificial Intelligence course, I have coded clustering and classification algorithms such as K-Means, K-Nearest Neighbour, Fuzzy K-Means, Maximum Likelihood Estimation, Least Squares Regression, Linear Discriminant Analysis and Principal Component Analysis (both kernel and non-kernel). I have also implemented neural network and regression approaches as part of a project to predict, with competitive accuracy, the result of a basketball match between any two NBA teams factoring in player form, team form, player synergies and team chemistry and past head-to-head results.

In my free time, I am also working on a personal project of using topic modelling along with either sequence-to-sequence LSTM like models or attention models for concise and grammatically accurate news article auto summarization. However, most pertinent to this project is my prior experience with time series data and the insights I gained while participating in the recently concluded Two SIgma Financial Modelling Contest on Kaggle where I tried out autoregressive as well as machine learning approaches such as ARIMA, decision trees and XGBoost.

RESUME LINK

https://drive.google.com/file/d/0BxMVBEaJ5LTZWGIwVmJCbjZldE0/view?usp=sharing

PROJECT DETAILS

Data Collection

I have trawled through several research papers, surveys and governmental organization websites online and have zoned in on two possible data sources that seem to offer promise and may be a good place to start with exploratory analysis.

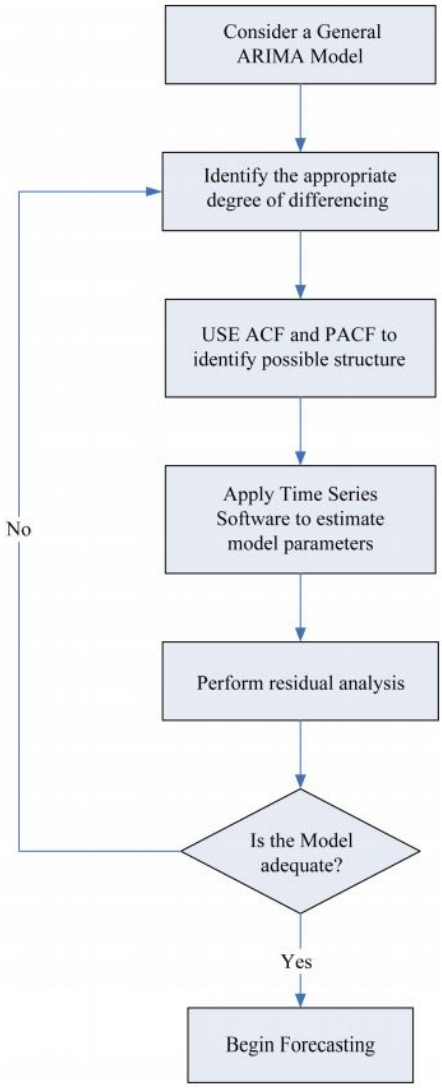
- https://www.fueleconomy.gov/feg/download.shtml
  This is a very extensive data source that contains data about electric as well as hydric vehicles. It has data on newer electric vehicle models upto 2017 and also drivetrain and other auxiliary information. However it only mentions distance and does not contain GPS coordinates and also has a problem of many missing values.
- http://www.chargecar.org/data
  This dataset contains GPS coordinates and altitude information which is very pertinent for our purposes. However the data and the models of electric cars observed are a bit dated.

Data Generation Through Simulation

Most of the recent datasets about electric and hybrid vehicle GPS data are proprietary and the public data in most cases is either old or inadequate. Thus a contingency plan must be thought of to generate more data. Note that in case a detailed enough and adequately large dataset is found sometime in the future, this phase of the project will be deemed surplus to requirements.

Following are some useful methods to generate realistic future values for already available time series data:

- **Discrete Event Simulation:**
  In order to determine the next event in a stochastic simulation, the rates of all possible changes to the state of the model are computed, and then ordered in an array. Next, the cumulative sum of the array is taken, and the final cell contains the number R, where R is the total event rate. This cumulative array is now a discrete cumulative distribution, and can be used to choose the next event by picking a random number between zero to R and choosing the first event, such the random number is less than the rate associated with that event.
  A probability distribution such as Bernoulli, Binomial, etc is used to describe the potential outcome of a random variable and limit the outcomes where the variable can only take on discrete values. Reaction methods and optimisations e.g. Next Reaction method and Log Direct method can then be used to simulate future values using the cumulative array.
- **Autoregressive Methods:**
  Establishing a model that describes the structure of an observed time series enables meaningful forecasts to be drawn from the model. One such widely used autoregressive method that operates this way is the ARIMA model. The whole pipeline is mentioned below.



The purpose of this process is to establish the underlying model that describes the time series through specifying the p and q parameters once the appropriate order of differencing d has identified a stationary process. The robustness of the generated model is evaluated through Autocorrelation and Partial Autocorrelation Function plots and once deemed robust enough, forecasting can begin.

- **Exponential Smoothing Methods:**

  The application the first order exponential smoothing method requires a choice of smoothing factor λ and the number of observations to smooth against. There is no analytical method to determine the optimum choice of smoothing factor and it is necessary to investigate various levels of λ and choose the smoothing factor that minimises the squared sum of the forecast errors e_t defined by

$$SS_E(\lambda) = \sum_{t=1}^{T} e_t^2 .$$

Similarly there is no method to determine the optimum number of observations to forecast against. However, we can use the fact that the influence of past the observations decays geometrically over time and the impact of the decay can be evaluated against a set of observations. The forecasts generated by the exponential smoothing and autoregressive process in practice, are remarkably close. The advantage of the exponential smoothing method lies in its simplicity of execution and also that it requires fewer data points to produce realistic forecasts. However, these methods suffer from the bias of assuming that the time series is stationary, which is not the case for the vehicular driving data that we will be dealing with.

- **Monte Carlo Simulations:**

  Systems or processes that can be modelled through an underlying probability distribution are open to simulation through the Monte Carlo method. The method simulates the behaviour of a system by taking repeated sets of random numbers from the underlying probability distribution of the process under investigation. A specific application of the Monte Carlo method is dependent on the nature of the underlying probability distribution of the system or process under investigation. However the method application is consistent and will follow the steps outlined below:

  - Define a distribution of possible inputs for each input random variable. This requires recognition of the underlying probability distribution of the process. This may be directly apparent or may require empirical observation of the process under investigation. This may be a problem because for electric cars, the inherent probability distribution is not known and we might not have enough data to infer it through artificial means.
  - Generate outputs randomly from those distributions by the selection of an appropriate random number generator to model the observed probability distribution.
  - Deterministically computing the desired output variable or variables from the generated random numbers.
  - Aggregate the results of the individual computations into the final result.

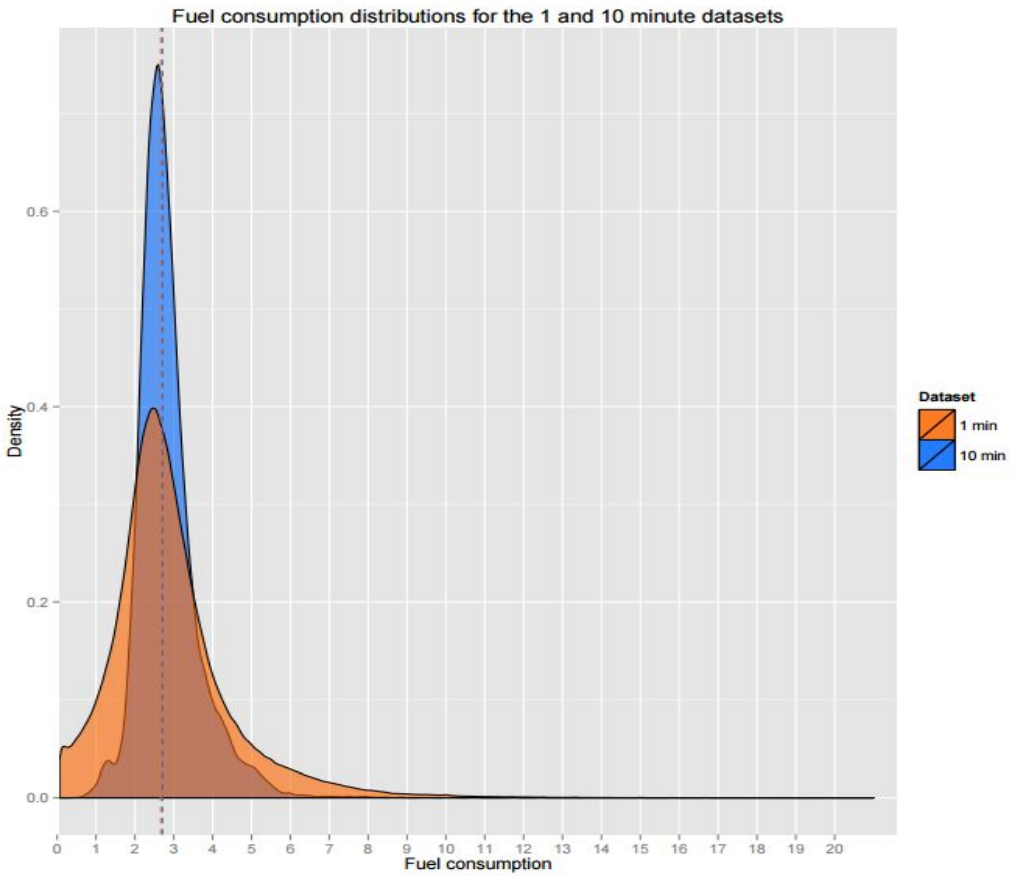
  One obvious advantage of using Monte-Carlo methods is that they make no implicit assumption about the presence or lack thereof stationarity in the time series.

Exploratory Data Analysis and Processing

Reading through multiple survey articles about similar energy consumption prediction approaches applied to conventional vehicles, it becomes apparent that initial processing of the

available data plays a vital role in the success of the learning algorithms. A few useful lessons from those surveys that we could do well to heed are:

- Several datasets contain attributes such as driver information, etc which are generally superfluous and either have no effect on the predictive performance or affect it negatively. Such irrelevant attributes must be found out and eliminated.
- The input data sampling rate can affect the final predictive performance significantly. Having too high a sampling rate can negatively affect predictive performance as it incorporates a lot of noise and increases the time series volatility causing learning algorithms to suffer. The figure below shows heightened variance when the sampling rate is 1 minute vs when it is 10 minutes, as computed on a dataset of the Swedish Transportation Authority.



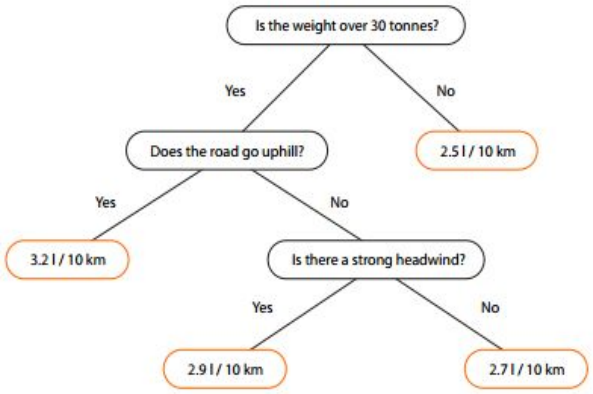
- One could also do analysis of vehicle characteristics, look for correlations and use statistical methods such as pca to reduce them to a few key metrics for use in future models.

Formulation of a Learning Approach

There exist a large selection of Machine Learning Approaches that can be and have been promisingly used for range estimation and fuel consumption prediction of electric vehicles. A

good place to start would be linearized regression as it is simple and can serve as a baseline. The simpler the possible solution, the better. However, during the course of this project. it is also important to explore certain techniques which lend themselves particularly well to time series regression and prediction, some of which are detailed below:

- **Classification and Regression Trees (CART):**
  Decision trees are a simple method of supervised learning in which the final model takes a vector of attributes as inputs and returns a single value, or decision, as output. In a decision tree, leaf nodes represent the decisions and branches represent conjunctions of attributes that lead to those decisions. A regression tree is a modified version of such a decision tree which is used for regression, wherein the leaf nodes can take up continuous values. Effectively, regression tree is a tree of nodes where each leaf node has a linear function of some subset of numerical attributes, rather than a single value which is the case for classification trees. An example is shown below.



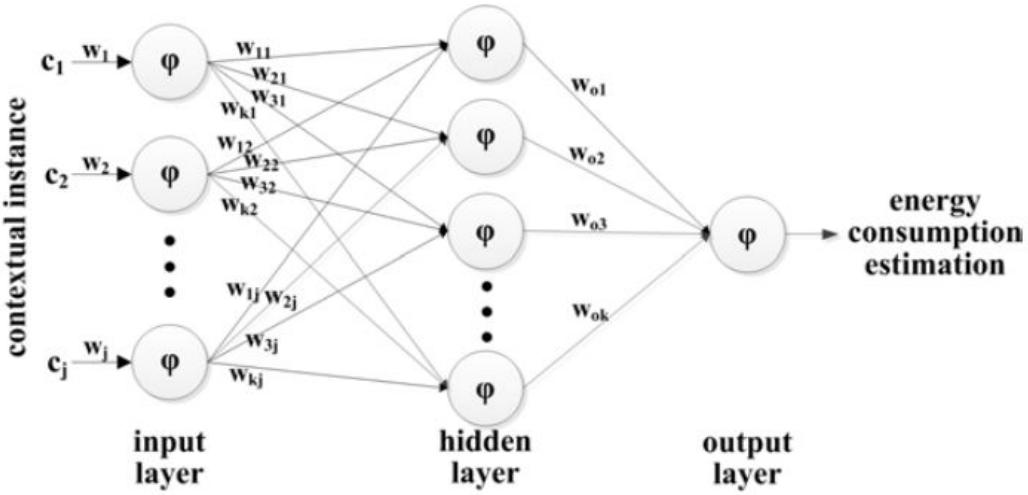
  The order in which to place the nodes and which node to choose as the root is decided by examining the entropy and information gain of the attributes. Information gain is the expected reduction of entropy achieved after eliminating an attribute from the equation.
- **Random Forest:**
  A random forest for regression is an ensemble learning method where several regression trees are trained and which outputs the mean prediction of the individual trees Instead of making the prediction based on one tree, it depends on a collection of trees to take the decision. Being different from other bagging techniques, RF adds an additional layer of randomness to bagging. Similar to other bagging models RF also constructs each decision/regression tree using a bootstrap of sample data. However, the tree building procedure is different. Random forests use a modified tree learning algorithm that selects a random subset of the attributes at each candidate split in the learning process. Random forests compensate for the tendency of decision trees to overfit to training data.
  Moreover, facilitating the estimation of variable importance and outlier detection are other benefits of this algorithm. Furthermore, RF is reasonably fast to obtain and can be easily

parallelized. A fine-tuned version of RF can be obtained by backward-elimination of predictors based on the given variable importance.

Random forests uses two parameters for tuning a model fit. They are features per tree and number of trees. The default number of features per tree is usually set to the square root of the total number of features and number of trees is usually selected to be as high as possible while keeping training time reasonably short. In order to find optimal parameter settings I plan on iterating over various values of number of features per tree, fit a model to the data and evaluate the RMSE of the fitted model. An appropriate value would be one that keeps both the RMSE and model training time low.

- **Artificial Neural Network (ANN):**

  A Multi Layer Perceptron (MLP) architecture can be explored wherein structure of the designed network is depicted in the figure below and consists of the input layer, one hidden layer and the output layer.



According to the MLP network structure, the representation for the target function can be written in the following nested form:

$$\hat{f}(\vec{c},\vec{w}) = \phi\left(\sum_k w_{ok}\phi\left(\sum_j w_{kj}\phi(w_j c_j)\right)\right)$$

where $\phi(.)$ is a sigmoid activation function, w_{ok} is the synaptic weight from neuron k in the hidden layer to the single output neuron o, w_{kj} is the synaptic weight from neuron j in the input layer to the neuron k in the hidden layer and c_j is the jth element of the input vector c. The sigmoid activation function $\phi(.)$ applied to the MLP network is:

$$\phi_j(u_j(n)) = \alpha \cdot tanh(b \cdot u_j(n)), \qquad (a,b) > 0$$

The design of the learning system is completed with the adoption of a learning algorithm. The most widespread choice in case of MLPs is the backpropagation algorithm, which searches the space of possible hypotheses using gradient descent to iteratively reduce

the error in the network fit to the training dataset. The error criterion used can be something like an average square error as shown below.

$$E_{av} = \frac{1}{N}\sum_{n=1}^{N} E(n) = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{2} e_o^2(n) = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{2}(d_o(n) - y_o(n))^2$$

- **Gradient Boosting:**

  Gradient Boosting (GB) is another ensemble predictive boosting algorithm for regression and classification problems which achieves optimal prediction by minimizing a loss function. Similar to other boosting algorithms, GB builds the model in stages and generalizes them by allowing optimization of an arbitrary differentiable loss function. Different loss functions such as least square, least absolute deviation, and Huber-M loss function are used for regression.

  Carrying out variable selection during the fitting process can be recognized as a key feature of GB. Further GB algorithms provide prediction rules that have same interpretation as common statistical models. This becomes a major benefit of GB over other ML algorithms such as Random Forest, which provides non-interpretable black-box predictions.

  An example of Gradient Boosting that has been especially successful in the past few years is the Extreme Gradient Boosting implementation (XGBoost), which is an advanced implementation of GB which provides several significant improvements over the vanilla GB implementation:

  - XGBoost uses regularization to control overfitting even further.
  - XGBoost allow users to define custom optimization objectives and evaluation criteria.
  - Perhaps most importantly for the kind of data that we may find, XGBoost automatically learns how to deal with missing values by replacing them with imputations that minimise the error. The algorithm computes the default split direction at each node using the non-missing samples only and when a missing value is encountered, it is classified in the default direction.
  - XGBoost allows user to run a cross-validation at each iteration of the boosting process thus making easy to get the exact optimum number of boosting iterations in a single run rather than performing time consuming grid-searches for parameter tuning that regular GB models must perform.
  - Finally, the most significant improvement in XGBoost lies in the fact that it does not chase gains in predictive performance greedily and hence is unlikely to get caught in a local optimum of the hypothesis search space.

    XGBoost specifies a max split depth parameter beforehand and runs the model up to that depth irrespective of how it helps or hurts the final result. It then starts retrospectively pruning the tree backwards and removes splits beyond which

there is no positive gain.

This is very useful as sometimes a split of negative loss say may be followed by a split of positive loss. GBs would greedily stop on encountering the first split thus missing out on predictor of high discriminative power.

- **Support Vector Regression:**

  Support Vector Machines (SVM) are a technique commonly used with great effect in cases where supervised learning must be performed without any domain specific insight into the problem. In its original formulation SVMs do classification of data points by a maximum margin decision boundary. For example an SVM might find the line between two clusters of data points that give the largest margin to the clusters. To find such a decision boundary the SVM finds so called support vectors, which are the data points that lie on or inside the margin. Using the kernel trick and dual formulation of the svm optimization problem different kernels may be used to embed the input data in a higher dimensional space, producing non-linear classifiers and greatly expanding the hypothesis space.

  Support Vector Regression (SVR) is an extension of the SVM where the same principles are applied to do regression instead of classification. Instead of finding a decision boundary with maximum margin the SVR finds a function approximation that minimizes the error. Like SVMs, SVR optimizes the generalization properties of the model. They rely on defining a loss function that ignores errors which are situated within a certain distance of the true value, this distance is denoted by the variable . Thus they depend only on a subset of the training data and ignore any training data close to the model prediction. The dual formulation of the resulting optimisation problem is shown below.

$$\min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \frac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\intercal Q(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) + \epsilon\sum_{i=1}^{l}(\alpha_i+\alpha_i^*) + \epsilon\sum_{i=1}^{l} z_i(\alpha_i-\alpha_i^*)$$

$$\text{subject to}$$

$$\mathbf{e}^\intercal(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0,$$

$$0 \le \alpha_i, \alpha_i^* \le C, i = 1,\ldots,l,$$

$$\text{where}$$

$$Q_{ij} = K(\mathbf{x_i},\mathbf{x_j}) \equiv \phi(\mathbf{x_i})^\intercal\phi(\mathbf{x_j}).$$

$$\text{and}$$

$$\mathbf{e} = [1,\ldots,1]^\intercal$$

  Where, $\{(x_1, z_1), \ldots, (x_l, z_l)\}$ is the set of training points, $x_i \in R_n$ is a feature vector and $z_i \in R_1$ is the target output. Also the function $K(x_i, x_j)$ is the kernel function that allows mapping of the problem into higher dimensional spaces.

By virtue of past personal experiences in regressing over non-stationary time series data, I personally feel that the Random Forest and Gradient Boosting methods are most promising.

Gradient Boosting methods using Regression Trees (CART) as the individual learners such as XGBoost have been wiping the floor with the competition over the past few years.
That being said I feel it is necessary to evaluate all the approaches with respect to how well they work on a case by case basis. I am also open to discussing and implementing any other approaches that the mentors may have in mind as the project progresses.

PROJECT TIMELINE: PRE PROJECT PHASE

3rd April (21:30 IST)

Submit final draft of proposal.

4th April - 14th April

Discuss the various candidate ML approaches with the mentors along with their merits and demerits.
I will also use this time to gain domain specific insights from the mentors about the problem and the accompanying data, that they might have gained while previously working on a similar problem.

15th April - 4th May

End Semester Examinations. During this time I will be completely occupied in my syllabus work and thus would not be able to devote any time towards the project.

5th May - 30th May

I will spend this time to get a hang of the Green Navigation codebase and the internals so as to make things easier when the final integration of the Energy Consumption Machine Learning module takes place.

I will also use this time to decide upon a machine learning framework of choice, after discussion with the mentors, in which the exploratory analysis and the final implementation are to be carried out

PROJECT TIMELINE: GSoC PROJECT PHASE

The slated duration of the project is about 12 weeks. Given below is details about how I plan to split the work. I am not explicitly mentioning unit tests everywhere with the underlying assumption that they will be written as and when the corresponding functionality is implemented. However, the final integration tests are separately mentioned. The same also holds for

documentation of individual bits of functionality which will be done as soon as new capabilities are added.

Week 1

This phase will be about gathering as much data as possible regarding Electric Vehicle driving patterns, preferably ones with GPS coordinates and altitude information from which terrain aspects can be inferred. All the work regarding data cleaning and organizing is to be done during this phase.

It is also during this phase of the project that it will become clear whether the data at our disposal is adequate or not, depending on which the next phase will inspect the aforementioned time series simulation techniques so as to generate more data.

Week 2 - Week 3

The scant data that we will have at this stage will be separated into training, validation and testing sections which will then be used evaluate the strength of various time series simulation approaches with respect to our data.

The chosen method will then be used to augment the existing data and create a larger dataset on which various learning approaches will be explored thereafter.

Week 4 + Phase 1 Evaluation Deadline

This week will be used to test out the artificial neural net approach using multi layer perceptrons. This technique will be tested by separately tuning several aspects such as hidden layer size, activation function, as well as the sampling frequency of the input data.

Week 5

Catch-up and review week. Try and smooth over all the previously done work and write more extensive tests on top of the already existing ones. If all goes well up till this point and there is time to spare, then try and conjure up a stretch goal to be implemented after discussion with mentors.

Week 6 - Week 7

During this phase, Classification and Regression Trees (CART) and Support Vector Regression (SVR) techniques will be investigated as possible solutions.

For Regression Trees, a number of overfitting avoidance measures will have to be tested out such as the different types of pruning or early stopping.

For Support Vector Regression, a decision needs to be made over whether the underlying support vector machine is a hard or a soft one and which kernel suits our data the most.

Both these methods also need to be tested with input data of various sampling frequencies to ascertain which produces the best results.

Week 8 - Week 9 + Phase 2 Evaluation Deadline

During this phase, Random Forests (RF) and Gradient Boosting techniques will be investigated as possible solutions.

Random forest parameters can be tuned by doing a grid search over realistic ranges for appropriate values of number of trees and number of attributes per tree.

However, most of the work during this phase will go into fine-tuning the various aspects of modified GB models such as XGB. Some of the parameter to be tuned include:

- Sampling frequency of the input data
- Learning rate
- Subsampling rate deciding the fraction of the observed samples and attributes to be randomly picked for each weak learner
- Maximum tree depth
- Gamma node split threshold deciding the minimum loss reduction requires to make a split
- Choice of loss functions e.g. Logistic, SoftMax
- Choice of validation evaluation metric e.g. RMSE, LogLoss

Week 10

Catch-up and review week. Try and smooth over all the previously done work and write more extensive tests on top of the already existing ones. If all goes well up till this point and there is time to spare, then try and conjure up a stretch goal to be implemented after discussion with mentors or maybe start working on documentation of existing modules.

Week 11

After extensive testing of candidate approaches, a solution has to be chosen to be integrated into the Green Navigation functionality and the scope of its application in the system is to be decided and after the integration is completed, a scheme of integration testing will have to be chosen so as to be implemented in the following week.

Week 12 + Final Deadline

I will devote the last week to finishing up the work and testing the functionality after integration. Also any required accompanying documentation will have to be completed in this week.