**Performance Prediction**

by

Atharva Wandile

**Abstract**

The aim of the project is to predict the performance of students of a particular class, given their performance in the previous years as well as the performance of the previous batch.

The method used to achieve the result is linear regression algorithm trained by gradient descent optimizer. The project further evaluates and compares different learning rates and regularization parameters to obtain the optimal solution.

Source: https://github.com/atharvaw1/performance-prediction

**Acknowledgements**

I’d like to extend my thanks to Edumerge Pvt. Ltd. For giving me the opportunity to work as their intern and do this project. My special thanks to Mr Rajat and Mr Krishna Mohan for mentoring and guiding me through the internship.

# Index

1. Visualizing and Preparing the Data

2. Splitting Data and Feature Scaling

3. Training Data

4. Hyperparameter Tuning

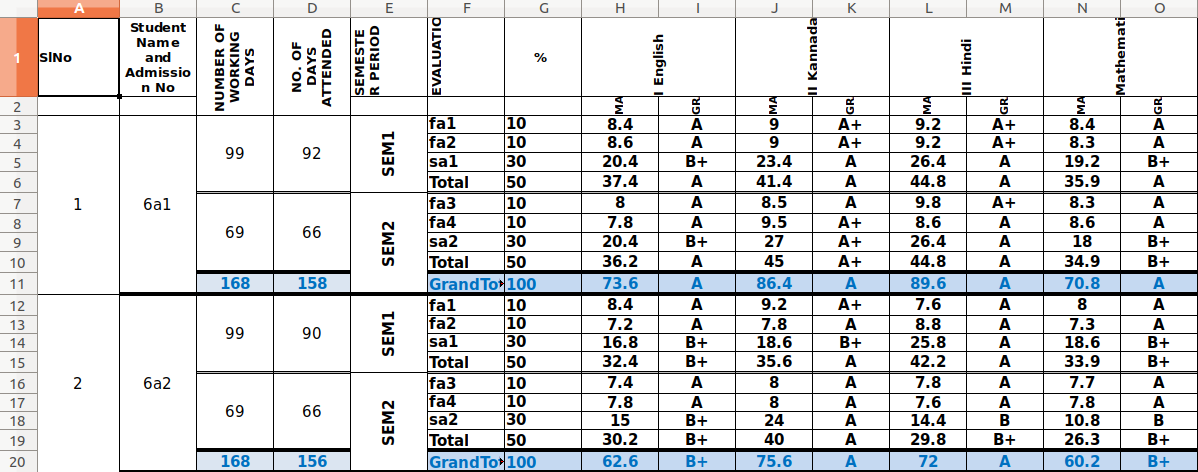
5. Prediction and Displaying Data

6. Further Scope for Studies

# Visualizing and Preparing the Data

# Data Extraction

First we need to see what the given data is and convert it into a suitable form for easy manipulation and calculations. The data is given in excel format.



Pandas is used to extract the data from excel into a pandas dataframe. Pandas allows for easy manipulation of the data. Later the pandas dataframe is converted to a numpy array which is accepted for calculations.

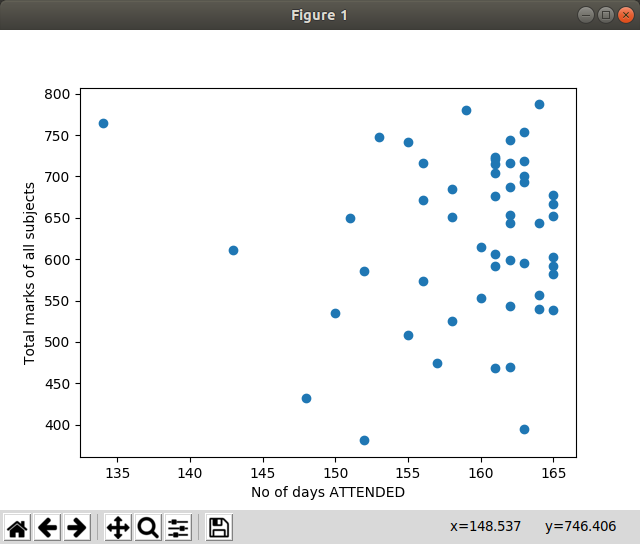
# Removing Unwanted Columns

The columns that are not required are removed. For eg. Student Id, Total, Semester, etc. have no consequence on the performance of the student.

These columns cannot be used for any analysis or prediction.

The grade column is removed as well as it is non-numeric and is based on the marks of the student which are already, given hence removing this does not have any effect on the performance of the model.

Next we check if the attendance has any correlation with the marks of the students



As seen from the plot above, the attendance has little consequence on the marks of the students. Hence the attendance is also removed.

# Removing Data Redundancy

It is important to remove redundant data to speed up the learning process and also to most importantly prevent overfitting in some cases.

The ‘total’ and ‘grand total’ for each student is the sum of all the individual tests only. It provides no new information and is therefore redundant.

Hence these rows are removed from the dataframe.

# Converting Dataframe Object into Numpy Matrix

Initially we used the pandas dataframe object to store the data as it is easy to manipulate and trasform. However for the machine learning we use tensorflow library which does not accept pandas dataframe as input. Therefore we need to convert the pandas dataframe into a numpy array. Pandas has an inbuilt function to convert the dataframe into a python matrix which is then converted to numpy array.

# Splitting Data and Feature Scaling

## Splitting Data

Now that the data has been converted into proper matrix format we need to split the data into the training and test sets.

Training set : The data that is used to train the model. The x as well as y data is given and the model trains the parameters according to the training set. This should be the main chunk of given data.

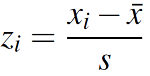
Test/Validation set : This data is used to test that accuracy of the model after it has been trained. This data is not seen by the model during training, hence it is similar to the real world data that the model needs to work on. This data is used to tune the hyperparameters of the model like learning rate, regularization parameter etc.

Before we split the data we need to make sure we are not picking the same data for training and testing every time as this can lead to bias. Hence we need to randomize the data set first. In order to randomize the data we first shuffle it and then split it into train and test sets. The extra students from each class are also dropped off at this stage ie. the students that are not present consistently in all the classes.

The splitting is done 80-20 that is 80% of the data is used for training and 20% of the data is used for testing.

## Feature Scaling

Once the data has been split we need to standardize the data. This brings all the different values of the features in the data to have mean 0 and standard deviation 1. This is important in order to achieve stable and fast convergence. Mean normalization has been used. In mean normalization we subtract each value with the mean and divide by the standard deviation.



Now the data is ready to be trained.

# Training the Data

## Declaring Variables

We first need to declare the different variables that need to be trained.

Wi  will be the weights of shape (1,no of features)

b is the bias variable which has same shape as W

Both these variables are initialized to zero at first.

## Hypothesis Function

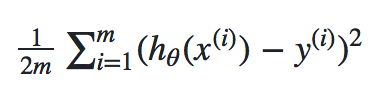
Now we need to form the hypothesis or prediction function.

The hypothesis is hθ = (Xi \*Wi) + b

This is our predicted outcome.

## Cost Function

The cost function is a measure of the error in the prediction compared to the actual values. The cost is calculated by the mean squared error.

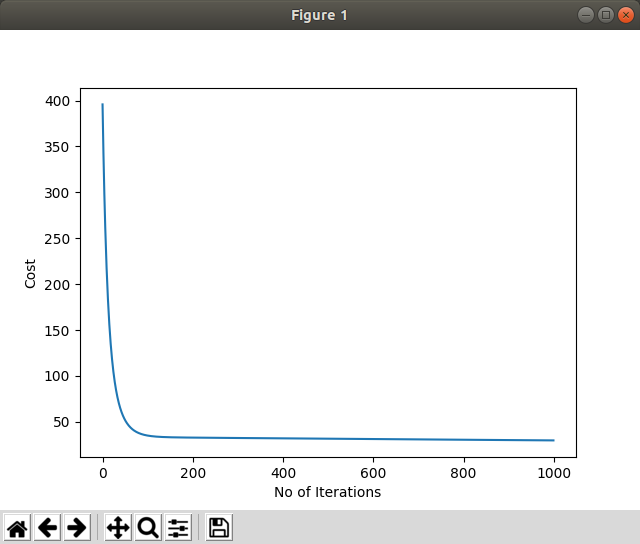


## Training Parameters

Now that we have the hypothesis and the cost function we need to train the parameters to fit our training data and minimize the cost function.

Gradient decent training algorithm has been used here. Gradient Decent is not implemented from scratch here however, the tensorflow inbuild function GradientDecentOptimizer() is used to train the parameters.

In order to check that Gradient Descent is working properly we plot the graph of No of iterations vs cost function. If the graph slopes down consistently then GD is working.



The above image is the graph for learning rate of 0.001 and 1000 iteration.

As seen the graph slopes downward smoothly. Hence we can conclude that the gradient descent is working.

# Hyperparameter Tuning

Now that the learning algorithm is setup the next step is to adjust the hyperparameters to get the optimized output.

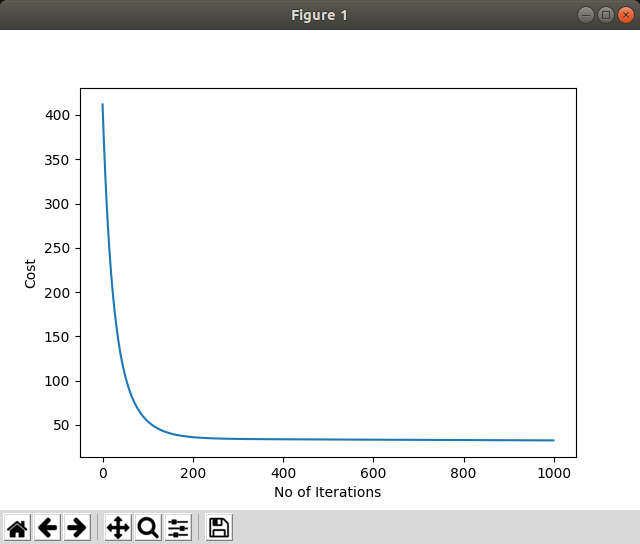
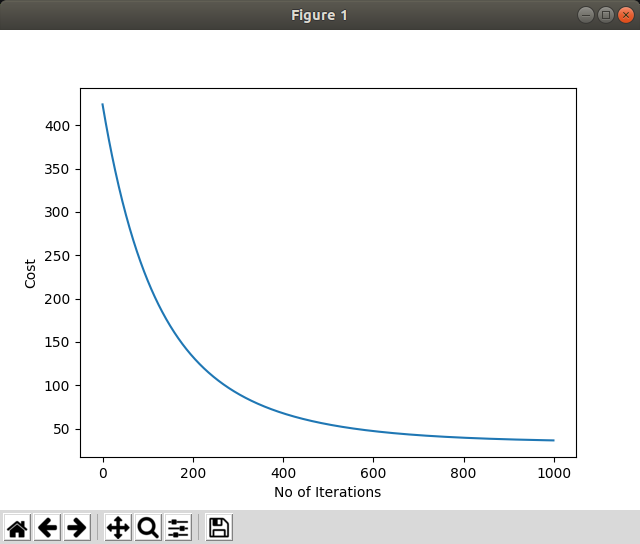
To tune the hyperparameters we need to train the data for multiple values of the hyperparameters and then check the cost for the test set for each. The values with the least cost will be chosen to train the data.

We will start out with a set of standard values for each hyperparameter and check the cost with each.

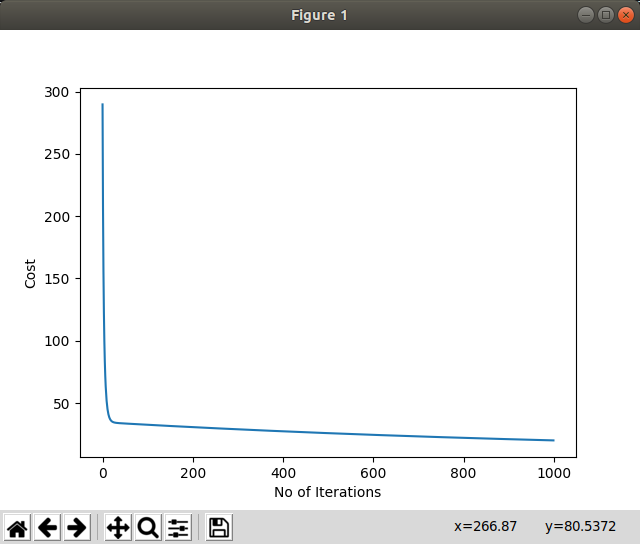
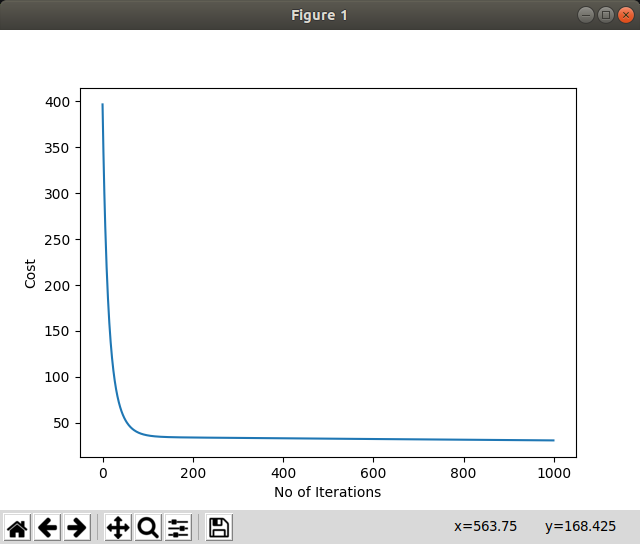
# Learning Rate

Standard Values : 0.0001,0.0005,0.001,0.005,0.01,0.05,0.1,1

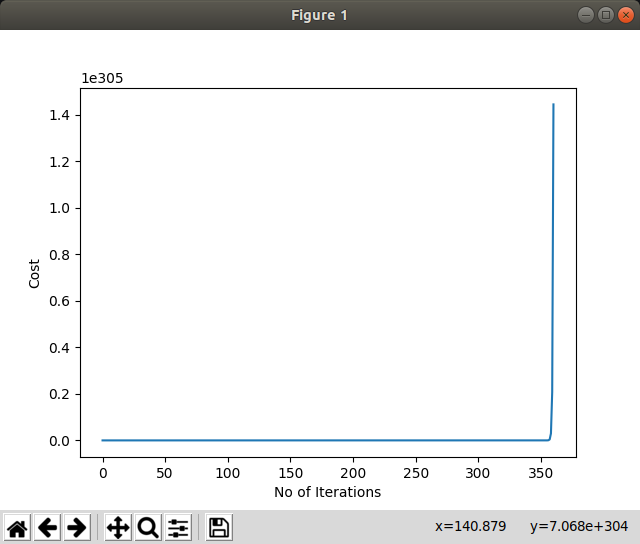
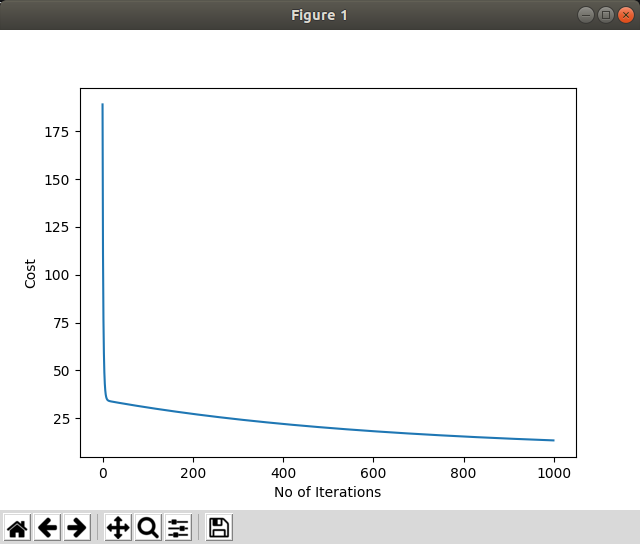
On training the model for each of these learning rates we get the following outputs



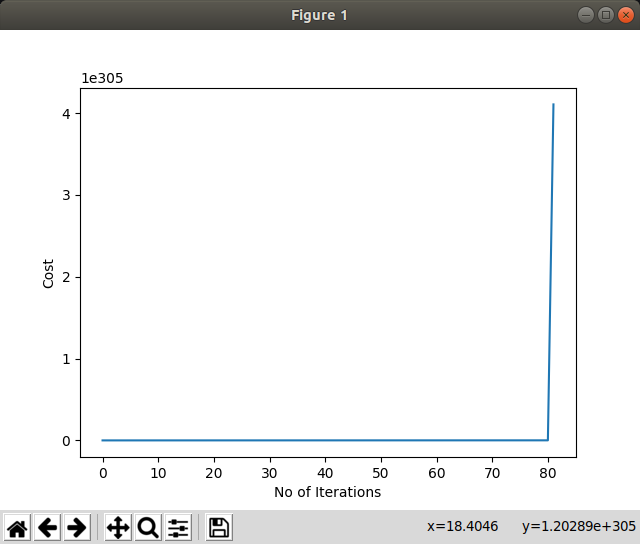
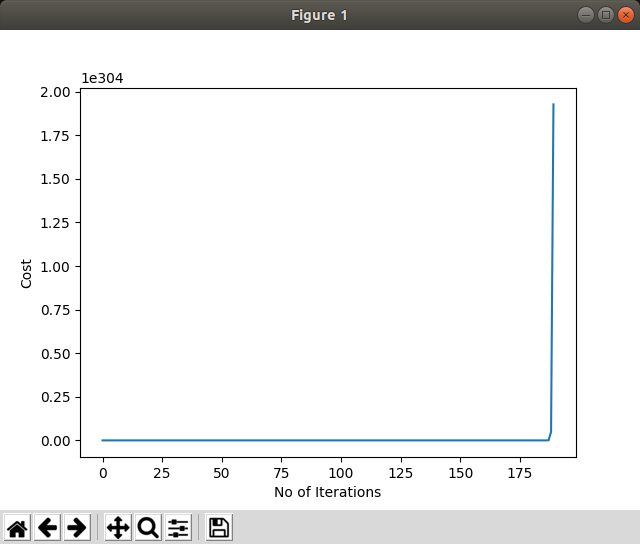
0.0001 0.0005



0.001 0.005



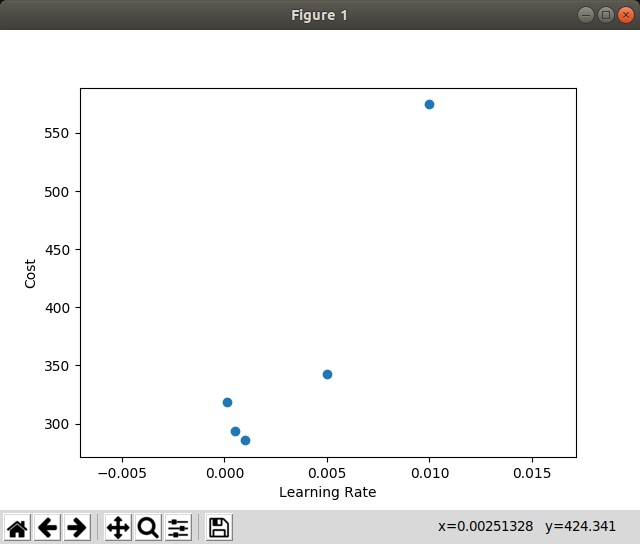
0.01 0.05



0.1 1

We can see that for the values 0.05, 0.1 and 1 GD has diverged. Hence we can immediately eliminate them.

The costs for each of these learning rates is as follows



Hence from the graph we can conclude that the learning rate of 0.001 is the best learning rate out of the given rates.

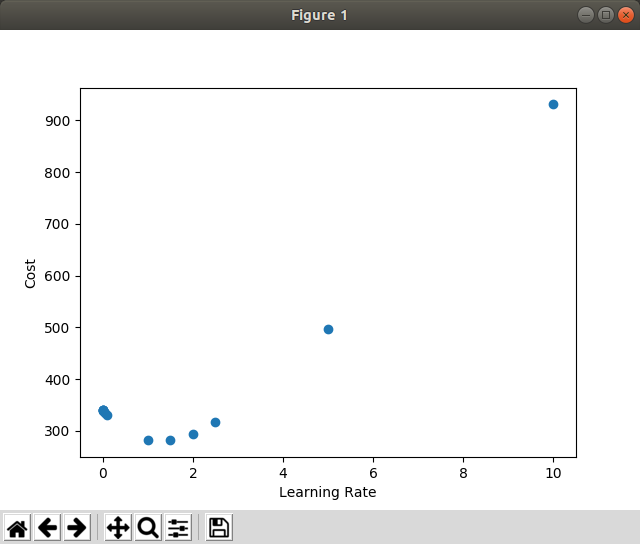
# Regularization

Regularization is used in order to prevent the model from overfitting the training data and have high bias. The regularization parameter decides how much we regularize the model. If it is too small then we risk overfitting the data, but if it is too large then we will have high variance in the model and will get a high degree of error. We need to choose the appropriate regularization parameter to get the optimum result. The procedure is similar to the picking of learning rate.

Standard Values: 0.0001,0.0005,0.001,0.005,0.01,0.05,0.1,1,1.5,2,2.5,5,10

The result obtained for the given standard values is

Costs: [340.42, 340.38, 340.32, 339.86, 339.30, 334.91, 329.72, 281.59, 281.66, 294.32, 316.19, 497.43, 930.58]



Clearly from the costs we can see that the minimum cost is 281.59 for the regularization parameter of 1. Hence we conclude that the optimal regularization parameter is 1.

# Prediction and Displaying Data

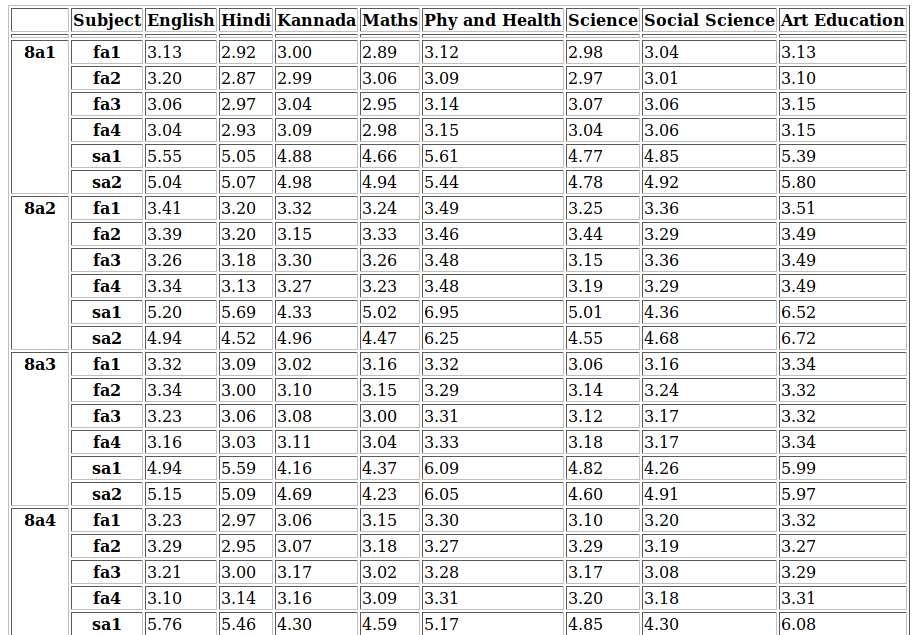
Now that the data is trained using the best hyperparameters we need to make a prediction for the scores of the students and display it in a suitable format.

Prediction can be done using the hypothesis or prediction function that has already been defined.

The cost that is obtained for the test data set is 378.55

The output is required to be in html format for better compatibility with Edumerge website.

First we convert the numpy matrix back into a pandas dataframe and convert it back into the input format. Next the dataframe can be directly converted into html page using pandas inbuilt function to\_html(). The result is stored in this html file.



# Further Scope for Study

The project has done some analysis of gradient descent however we can go further and test out the different types of gradient descent like mini batch GD, Stocastic GD etc. and these can be evaluated.

This project has only considered one of many training algorithms(Gradient Descent) used in the field of machine learning. Further studies can be made on different learning algorithms for eg. Normal Equation etc. and these algorithms can be compared at an empirical level.