Mounica Mandapati

A20151150

KICKSTARTER PROJECT SUCCESS

CS 5783 Project

**TABLE OF CONTENTS**

Introduction

Problem statement

Data Collection and Description

Data Cleaning

Feature Engineering

Exploratory Data Analysis

Modelling and Hyper parameter Tuning

Interpreting results

Conclusions

1. **Introduction:**

Crowdfunding has become one of the main sources of initial capital for small businesses and start-up companies that are looking to launch their first products. Websites like [Kickstarter](https://www.kickstarter.com/) and [Indiegogo](https://www.indiegogo.com/) provide a platform for millions of creators to present their innovative ideas to the public. This is a win-win situation where creators could accumulate initial fund while the public get access to cutting-edge prototypical products that are not available in the market yet.

Kickstarter is a community of more than 10 million people comprising of creative, tech enthusiasts who help in bringing creative project to life. Till now, more than $3 billion dollars have been contributed by the members in fueling creative projects. The projects can be literally anything – a device, a game, an app, a film etc.

Kickstarter works on all or nothing basis i.e. if a project doesn’t meet its goal, the project owner gets nothing. For example: if a project’s goal is $500. Even if it gets funded till $499, the project won’t be a success.

1. **Problem statement:**

At any given point, Indiegogo has around 10,000 live campaigns while Kickstarter has 6,000. It has become increasingly difficult for projects to stand out of the crowd. Of course, advertisements via various channels are by far the most important factor to a successful campaign. However, for creators with a smaller budget, this leaves them wonder,

*"How do we increase the probability of success of our campaign starting from the very moment we create our project on the Kickstarter website?"*

1. **Data Collection and Description:**

For the purpose of this project, data is collected from webrobots.io website.

This website has a scraper robot that crawls all Kickstarter projects and collects data in CSV format. The dataset contains 28 columns: 17 numeric, 21 categorical columns.

The description of the useful variables are:

1. Categorical Variables:

|  |  |  |
| --- | --- | --- |
| Variable Name | Levels | Explanation |
| id | 179,782 | Unique Identifier for Project |
| name | Text field | Name of the project |
| blurb | Text field | Project short description |
| category | Text field | Category of project |
| state | 4 | Describes state of project |
| country | 22 | Origin country of project |
| currency | 14 | Original currency of the projects |
| staff\_pick | 2 | Projects favored by Kickstarter Staff |
| spotlight | 2 | Check the project for Kickstarter spotlight |
| disable\_communication | 2 | Check the project for owner communication option |

1. Continuous Variables:

|  |  |  |
| --- | --- | --- |
| Variable Name | Range | Explanation |
| deadline | 05/03/2009- 03/03/2018 | Deadline of the project |
| launched | 04/24/2009- 01/02/2018 | Launch date of the project |
| created\_date | 04/21/2009- 01/02/2018 | Project creation date |
| state\_changed\_date | 05/03/2009- 03/03/2018 | Date of state change for the project |
| backers\_count | 1 - 105,857 | Total count of backers |
| static\_usd\_rate | 0.00877 - 1.7164 | USD conversion rate at the time of project inception |
| pledged\_amount | 0 - 10,266,845.74 | Total amount pledged towards project |
| goal\_amount | 0.01 - 100,000,000 | Goal amount set by project starters |

1. **Data Cleaning:**

Of the available projects, there were some anomalies in the data. So, data cleaning was done as follows:

1. There were around 20,000 duplicate project ids out of around 205,227 project ids. On observing that these duplicate project ids have the same column values, these project ids were removed from further analysis.
2. The dates like created\_at, deadline etc. are in interval format. So, they are changed to date format using ‘to\_datetime’ function.
3. The columns friends, is\_backing, is\_starred, permissions are having all missing values. So, they are dropped from further analysis.
4. The column is\_starrable is dropped since it has all 'false' values only. The columns photo, profile, source\_url since it is not useful since they don’t have any useful information.
5. There are 4 levels in target, but there are only 2 levels (i.e. successful, failed) in target that are useful for the analysis. So, removed other two levels (i.e. canceled, live) from further analysis.
6. There are 6 projects which do not contain project name. So, these rows are removed from analysis.
7. **Feature Engineering:**

With the available variables, below variables are created. These variables are also included in further analysis to improve model performances.

1. Categorical Variables:

|  |  |  |
| --- | --- | --- |
| Variable Name | Levels | Explanation |
| Creator\_name | Text field | Project creator |
| Main\_category | 15 | Category of the project |
| country\_us | 2 | Checks if the country if USA |

1. Continuous Variables:

|  |  |  |
| --- | --- | --- |
| Variable Name | Range | Explanation |
| creator\_projects\_count | 1 - 68 | Number of projects created by the project creator |
| days\_btw\_create\_launch | 0 - 3190 | Number of days between project creation and launch date |
| days\_btw\_launch\_stchange | 1 - 97 | Number of days between project launch and state  changed date |
| campaign\_length | 1 - 97 | Number of days between project launch date and deadline |
| blurb\_length | 1 - 151 | Length of project blurb |
| name\_length | 1 - 85 | Length of project name |
| blurb\_words | 1 - 43 | Number of words in project blurb |
| name\_words | 1 - 27 | Number of words in the project name |

Apart from all these variables, there were several text topics created from the project\_name and project\_blurb text field. The following rules were followed while creating text topics:

1. Minimum occurrence of word is 3
2. Removed English stop words
3. Converted all words to lowercase
4. Removed any characters other than alphabets and numbers
5. Removed any word with length less than 3
6. Maximum number of unique words is 300

On the project\_name field, built 5 meaningful text topics and on project\_blurb filed, built 20 meaningful text topics. Also, created another column which shows the dominant text topic for project\_name and project\_blurb for each project.

Example Text topic fields:

|  |  |
| --- | --- |
| Variable Names | Explanation |
| Blurb\_topic1 | Probability of first Text topic from the project blurb field |
| Dominant\_blurb\_topic | Dominant Text topic from the project blurb field |
| name\_topic1 | Text topic from the project name field |
| Dominant\_name\_topic | Dominant Text topic from the project name field |

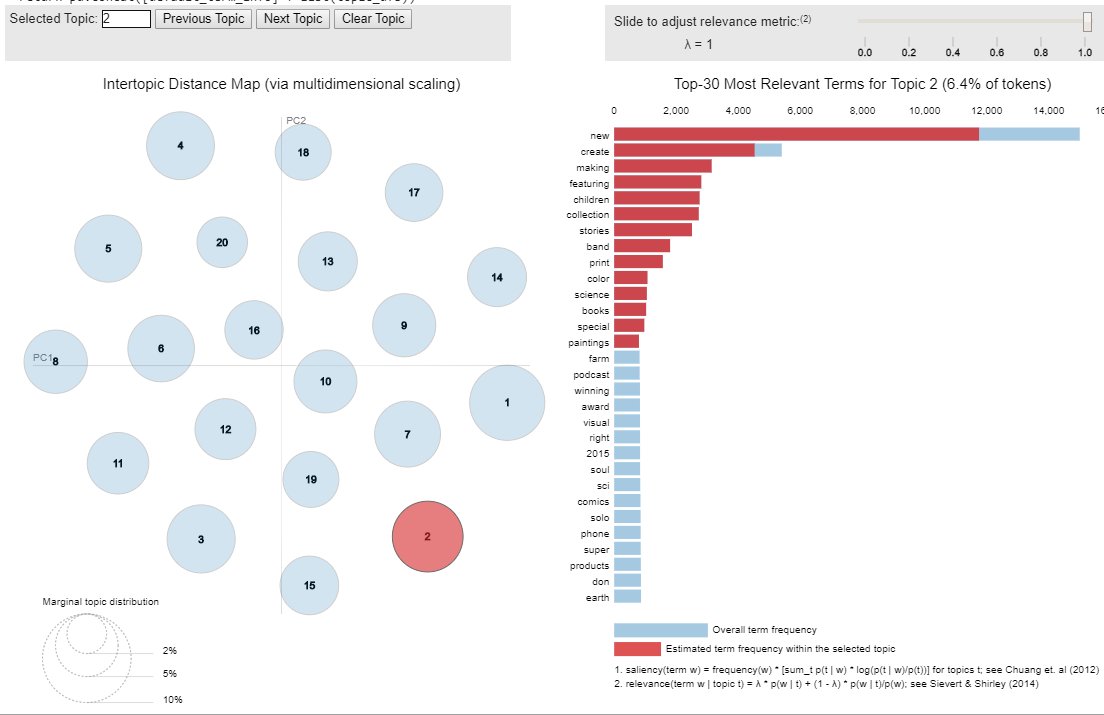
1. **Exploratory Data Analysis:**

Below are the descriptive statistics of the variables available in the dataset.

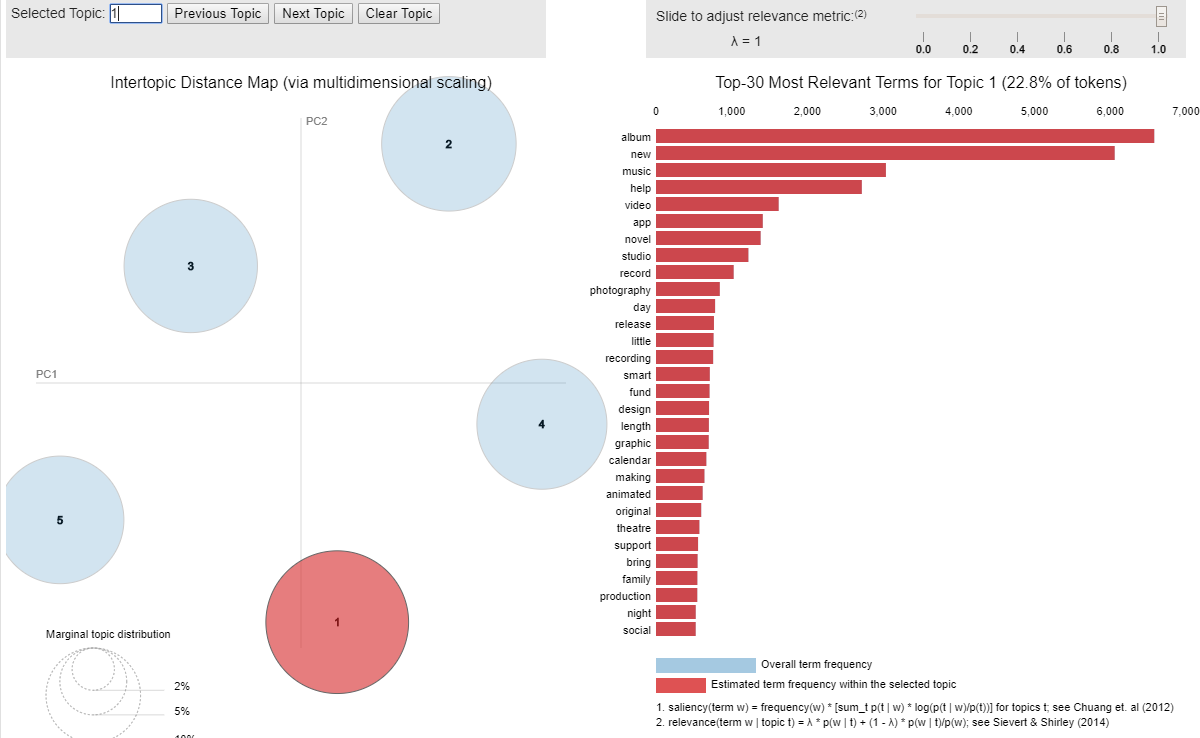
1. The intertopic distribution map:

It presents an overall view of the topic model. Different topics are plotted as circles, where overall prevalence was calculated as the areas of the circles. The centers of each topic were determined by computing the distance between topics; multidimensional scaling was used to represent the intertopic distances on a two-dimensional plane. PC1 indicates the transverse axis and PC2 indicates the longitudinal axis.

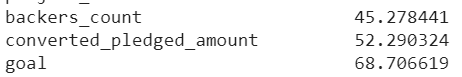
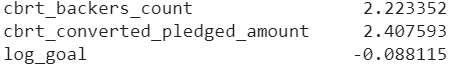
The bar chart in a descending order of the top 20 most useful terms, for interpreting a topic. The overlaid bars represent a given term’s corpus-wide frequency and the topic-specific frequency. In this bar chart, it is representing the top 20 useful terms of project\_blurb topic 2



Similarly, for the project\_name the intertopic distance map and top 20 useful terms for topic 2 are:



1. There are interval variables with higher skewness. So, transformed them accordingly using cube root and log transforms to lower their skewness.

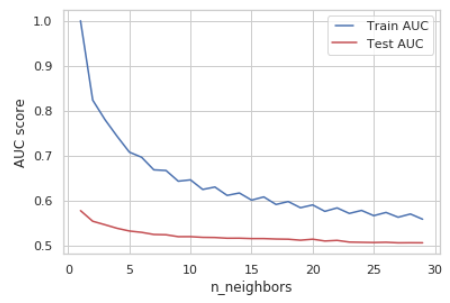
1. Checked the correlation among features and removed the features with multicollinearity.
2. **Modelling and Hyper parameter Tuning:**

The dataset after feature engineering and feature selection has 23 variables. Using all the 23 variables, built 4 machine learning techniques to predict the success of the Kickstarter projects.

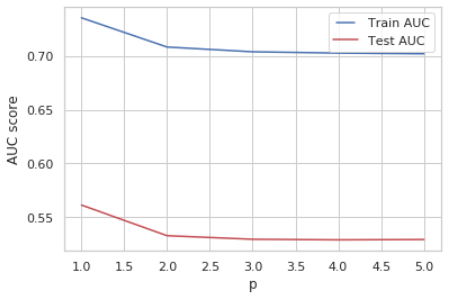
1. **K-nearest neighbor (KNN):** KNN is a distance based algorithm. It assumes that similar data points are near to each other. This algorithm is implemented using KNeighborsClassifier function from sklearn library.

By building KNN with default parameters, the model AUC and misclassification rate are 0.53 and 0.45. Keeping this as baseline model and tuned hyper parameters like n\_neighbors (number of neighbors), p (power parameter for the Minkowski metric)

**Tuning n\_neighbor:** On tuning the number of neighbors from 1 to 30, it is observed that the AUC decreases and making better predictions. But at the same time it is also overfitting as the number of neighbors is increasing.



**Tuning p:** On tuning the power parameter for the Minkowski metric from p=1 to 5, it is observed that p=1 is performing better on this data. When p=1, this is equivalent to using manhattan\_distance (l1), and euliddean\_distance (l2) for p=2. For arbitrary p, minkowski distance (l\_p) is used.

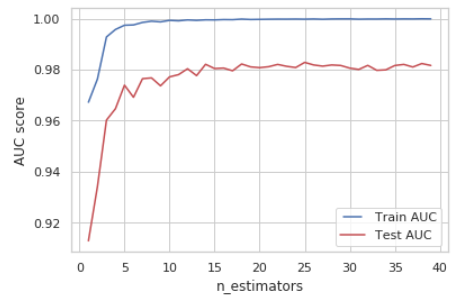


1. **Random Forest Classifier**

By building Random Forest with default parameters, the model AUC and misclassification rate are 0.97 and 0.02. Keeping this as baseline model and tuned hyper parameters like n\_estimators (number of estimators), max\_depth (maximum depth in a forest), min\_samples\_splits (minimum number of splits), min\_samples\_leafs (minimum number of leafs), max\_features (maximum number of features)

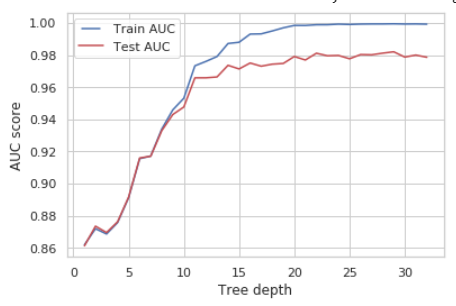
**Tuning n\_estimator:** n\_estimators represents the number of trees in the forest. Usually the higher the number of trees the better to learn the data. However, adding a lot of trees can slow down the training process considerably, therefore we do a parameter search to find the sweet spot.

On tuning the n\_estimators from 1 to 40, the test auc is highest at 15 estimators. After that it is almost same.



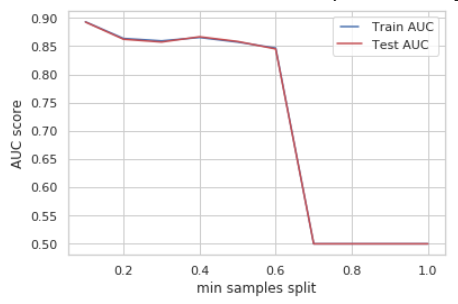
**Tuning max\_depth:** max\_depth represents the depth of each tree in the forest. The deeper the tree, the more splits it has and it captures more information about the data. We fit each decision tree with depths ranging from 1 to 32 and plot the training and test errors.

We see that our model overfits for large depth values. The trees perfectly predicts all of the train data, however, it fails to generalize the findings for new data

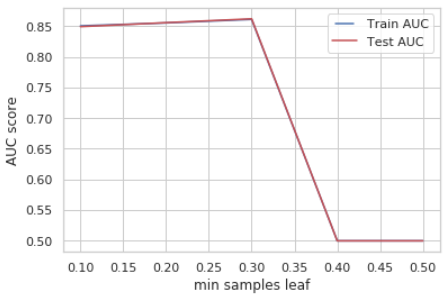


**Tuning min\_samples\_split:** min\_samples\_split represents the minimum number of samples required to split an internal node. This can vary between considering at least one sample at each node to considering all of the samples at each node. When we increase this parameter, each tree in the forest becomes more constrained as it has to consider more samples at each node. Here we will vary the parameter from 10% to 100% of the samples.

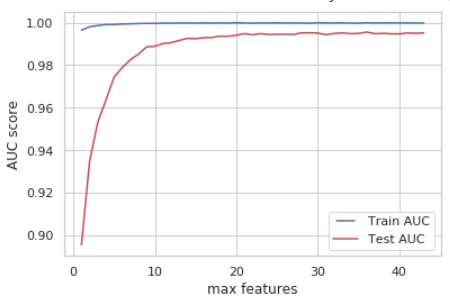
We can clearly see that when we require all of the samples at each node, the model cannot learn enough about the data. This is an underfitting case.



**Tuning min\_samples\_leaf:** min\_samples\_leaf is the minimum number of samples required to be at a leaf node. This parameter is similar to min\_samples\_splits, however, this describe the minimum number of samples of samples at leafs, the base of the tree. Increasing this value can cause underfitting.

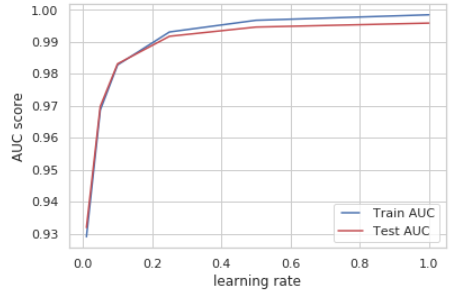


**Tuning max\_features:** max\_features represents the number of features to consider when looking for the best split. This is also an overfitting case. It’s unexpected to get overfitting for all values of max\_features. However, according to sklearn documentation for random forest, the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than max\_features features.

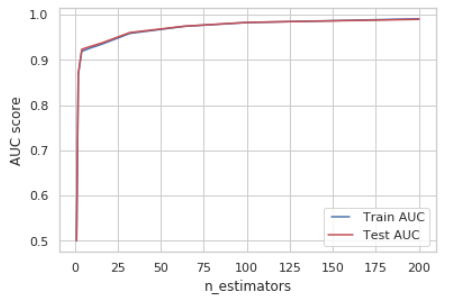


1. **Gradient Boosting:** baseline auc is 0.98 and misclassification rate is 0.014

**Tuning learning rate:** learning rate shrinks the contribution of each tree by learning\_rate. We see that using a high learning rate results in overfitting. For this data, a learning rate of 0.1 is optimal.

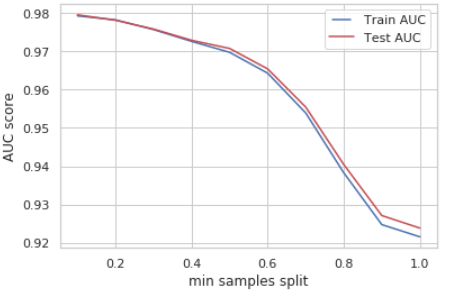


**Tuning n\_estimators:** Increasing the number of estimators may result in overfitting also. In our case, using 32 trees is optimal.

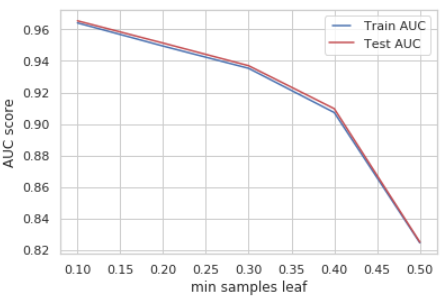


**Tuning max\_depth:** We see that our model overfits for large depth values. The tree perfectly predicts all of the train data, however, it fails to generalize the findings for new data

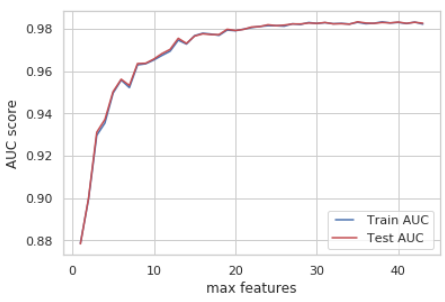
**Tuning min\_samples\_split:** We can clearly see that when we require all of the samples at each node, the model cannot learn enough about the data. This is an underfitting case.



**Tuning min\_samples\_leaf:** Increasing this value can cause underfitting.

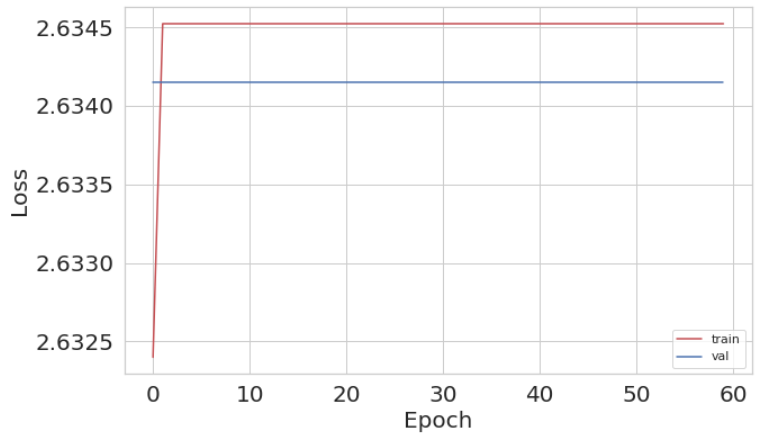


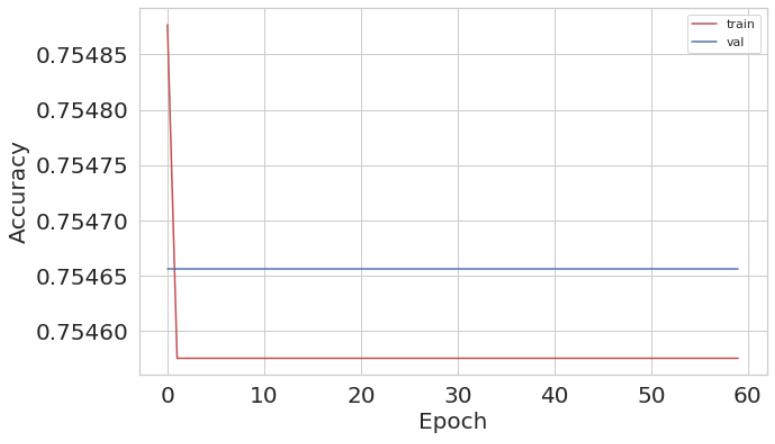
**Tuning max\_features:** Increasing max\_features to consider all of the features results in an overfitting in this case. Using max\_features = 6 seems to get us the optimal performance.



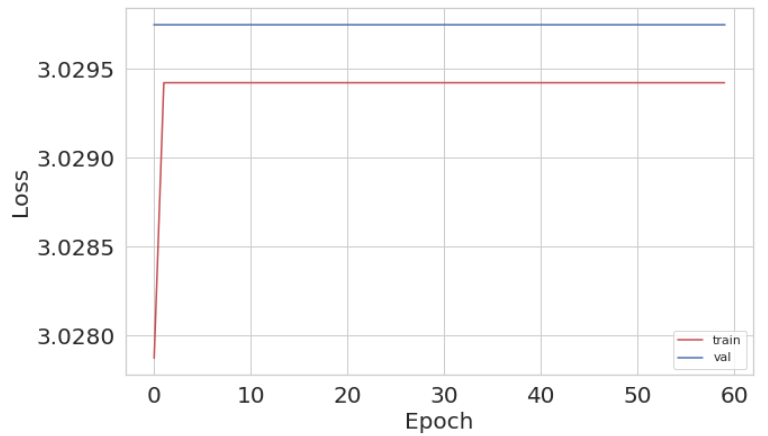
1. **Deep Neural Net (DNN):** One of the most common optimization algorithms is Stochastic Gradient Descent (SGD). The hyper parameters that can be optimized in SGD are learning rate, momentum, decay and nesterov. Learning rate controls the weight at the end of each batch, and momentum controls how much to let the previous update influence the current weight update. Decay indicates the learning rate decay over each update, and nesterov takes the value “True” or “False” depending on if we want to apply Nesterov momentum.

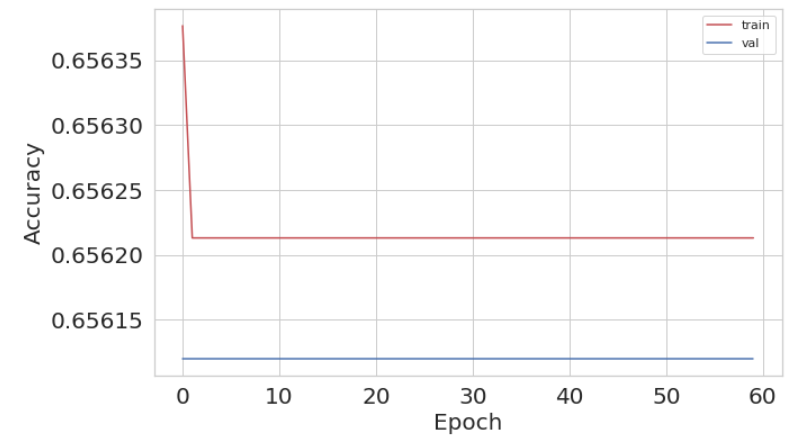
Typical values for those hyper parameters are learning rate=0.01, decay=1e-6, momentum=0.9, and nesterov=True. With these parameters built a 2 hidden layer neuron.



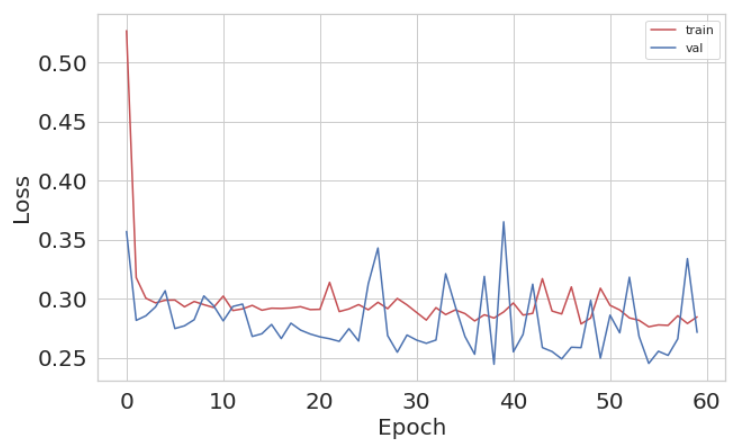


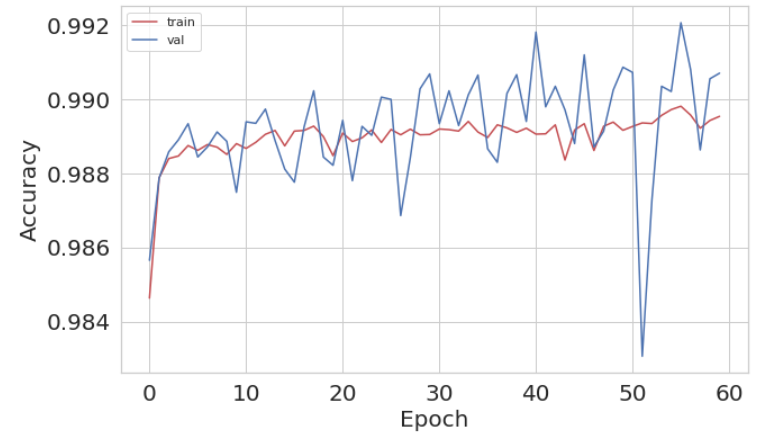
**Applying custom learning rate:** exp\_decay function is added to the above neuron along with few callbacks. After applying this custom learning rate, the validation loss has increased and accuracy decreased. Even though, the learning rate is smooth, this method did not prove to improve the performance for this data.





**Choosing an Optimizer:** Added an RMSprop Optimizer to above model, the results show that there is fluctuations in the results. But loss has drastically decreased and accuracy has improved a lot. So, RMSProp optimizer is performing better than SGD optimizer.





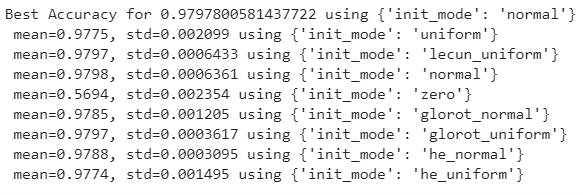
**Varying batch size:** On varying the batch size, it is observed that as batch size increases, the accuracy decreased.

|  |  |  |
| --- | --- | --- |
| **Batch size** | **Test accuracy** | **Test loss** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

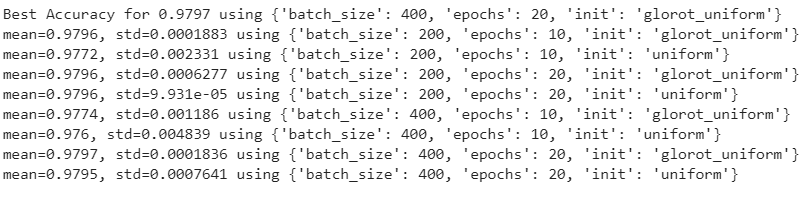
**Varying epochs:** On varying epochs, it is observed that, accuracy decreases with increase in epoch.

|  |  |  |
| --- | --- | --- |
| **Epochs** | **Test accuracy** | **Test loss** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Varying weight initializations:** Keras has different weight initializations like, lecun\_uniform, uniform, normal, zero, glorot\_uniform, he\_normal etc. All these initializations are tried on this data. For this data, the ‘uniform’, ‘glorot\_uniform’, ’lecun\_uniform’ are performing better with 0.9797 accuracy.



**Grid search Optimization:** On using the above best performing parameters with Grid Search, the best accuracy is for batch\_size = 400, epochs = 20, initialization = ‘glorot\_uniform’ .



1. **Interpreting results:**
2. **Conclusions:**