

A decorative graphic on the left side of the slide consisting of two overlapping parallelograms. The front one is blue and the back one is a light green. They are positioned diagonally, with the blue one partially covering the green one.

Model Analysis

Pragya Prakash
Niresh Anderson
Shravika Mittal



Ensemble Methods

We used the following ensemble methods to obtain better results:

- Random Forests
- Light GBM

Other approaches

- Neural Networks
- SVM
- LASSO Regression



What is Light GBM?

Currently the de-facto standard for most deep learning contests on kaggle, we present an in-depth analysis of why Light GBM works better than any other method we tried, and how we fine-tuned the parameters to obtain the best model.

Light GBM uses tree-based learning along with gradient boosting methods. Instead of increasing width of trees, it focuses on increasing number of leaf nodes in a tree.

It is gaining great popularity because it performs efficiently on huge amounts of data, and training requires less time and doesn't hog memory. It also supports GPU learning.



Why did we use Light GBM?

We had 9 lakh data points, each having 71 features after extensive feature extraction and analysis. We needed to perform regression to predict the log of transaction revenue generated per user.

We saw that even with a naive implementation of the Light GBM Regressor, there was a significant improve in rmse scores.



How does it compare?

Model Used	RMSE on Train data	RMSE on Test data
Baseline - LASSO	1.8159	1.8273
SVM	1.6912	1.7400
Random Forest Regressor	1.6114	1.6390
Neural Network (MLP)	1.6029	1.6282
Light GBM	1.5265	1.5812



Results

Features = 23

Training Complete

Training RMSE

RMSE is 1.515231599255304

Testing RMSE

RMSE is 1.586252713161361



With One hot encoding = only 13 features

Features due to one-hot encoding = 25

Training Complete

Training RMSE

RMSE is 1.5573015142720767

Testing RMSE

RMSE is 1.6156972183582672



Reasoning

Even a naive implementation of Light GBM performs much better because:

- Main reason is that using leaf-wise splitting rather than level-wise splitting is great at improving performance because, as each leaf is split more and more, the decisions focus more on features and the dependency and effect of important features on the learning is more.
- So, when growing on the same leaf in LGBM, leaf-wise algorithm reduces more loss than level-wise.



Issues with LGBM

However, Light GBM has it's own fair share of problems:

- Biggest problem would be that of overfitting. With the trees increasing in depth, the decision criteria would become quite complex and inadvertently overfit on the training data. This is an unsolvable issue with small datasets, but with the amount of data we have, it seems unlikely.
- There are more than 100 parameters, so quite difficult to fine-tune to get the best model.

Parameter Tuning

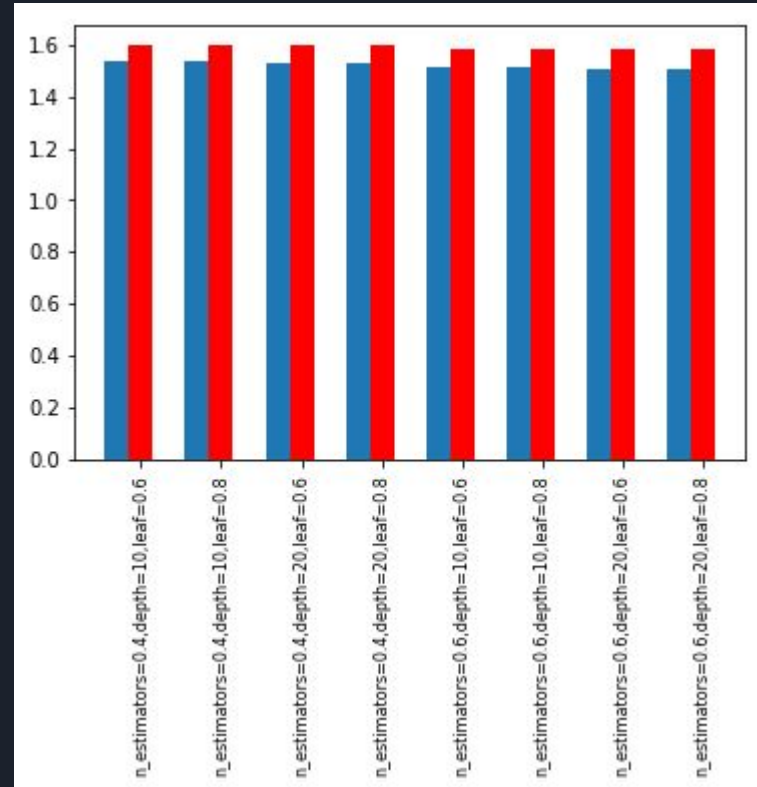
Applied GridSearch CV for parameter tuning. The most important parameters that affected the performance significantly were max_depth, number of leaves, bagging fraction, feature fraction. Using GridSearch, we obtained best parameters as follows:

Max-Depth = 15

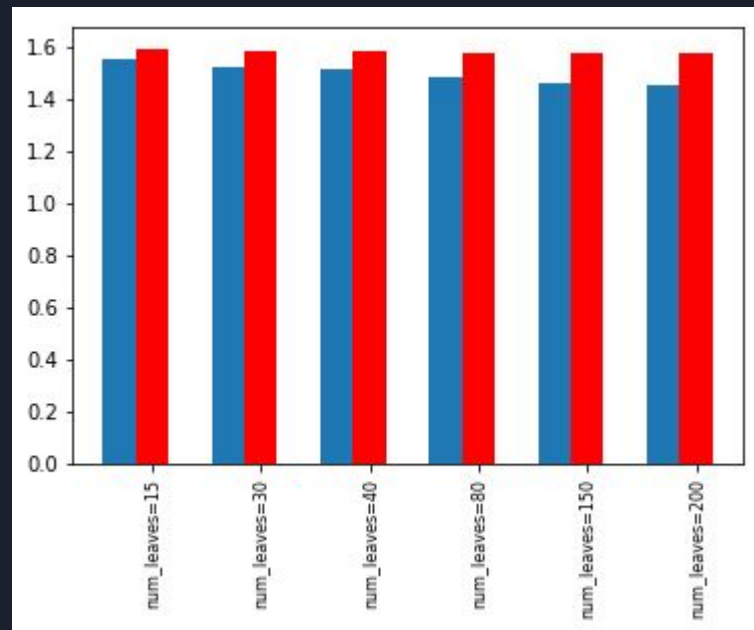
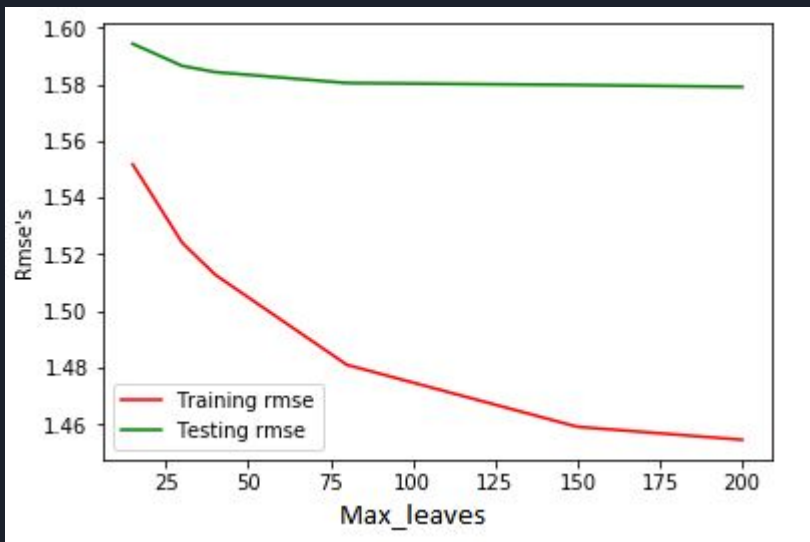
Num of leaves = 40

Feature fraction = 0.6

Boosting fraction = 0.8



Proof of perfect-fit claim





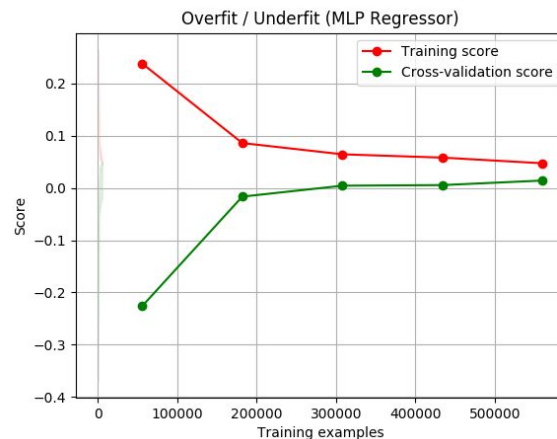
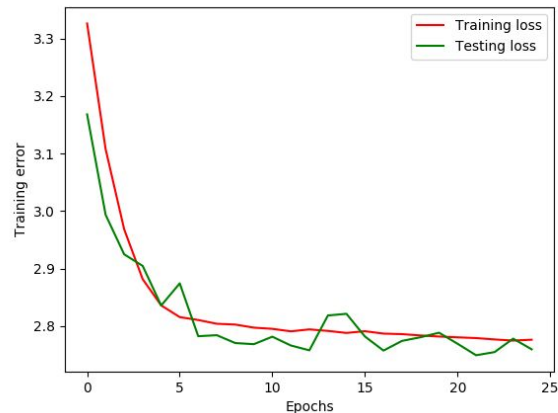
Neural Network - Architecture

- 3 layer network with 10 neurons each
- ReLu activation function
- Mean Squared Error loss function
- RMSprop optimisation

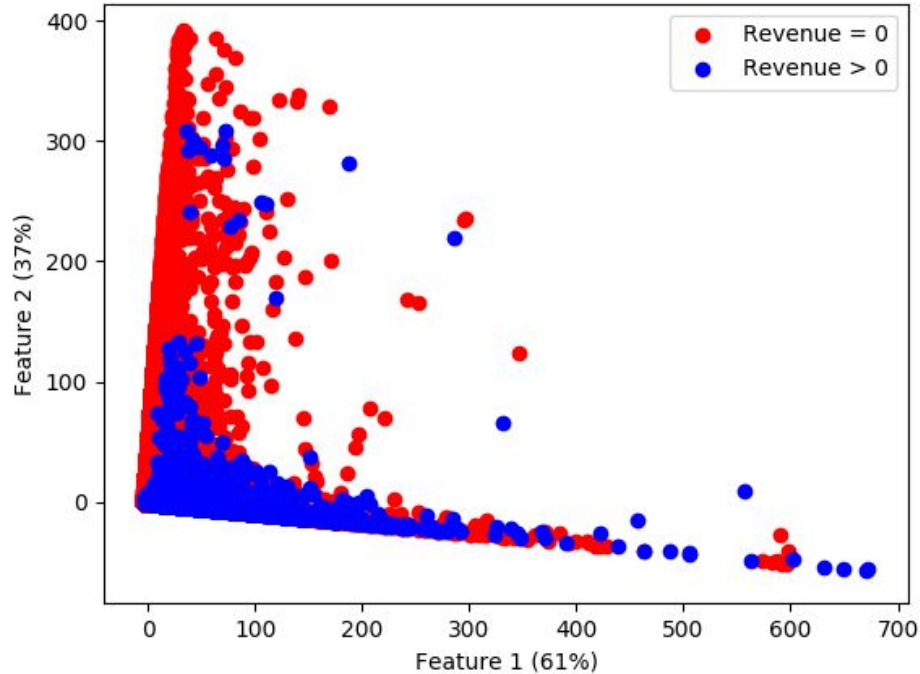
Learning Curve

- Early stopping has been used to minimise test error
- Model generalises well
- Train and test error are comparable

Note: Training error here is MSE



Data Distribution - PCA Visualisation





Why does this architecture work?

- PCA shows that the whole of the variance in the data is captured by just 2 dimensions (Fig 1 - previous slide).
- Problem boils down to solving a 2D regression problem
- EDA shows that the transactions that generate revenue and the ones which don't, are moderately well separated
- Therefore, a neural with just 3 layers and 10 neurons each works better than complex architectures



Ablative Analysis

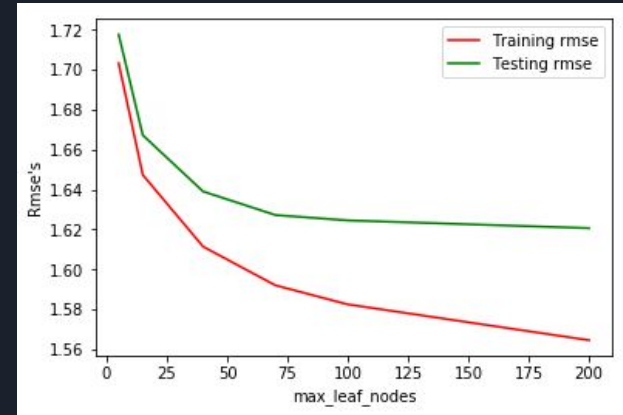
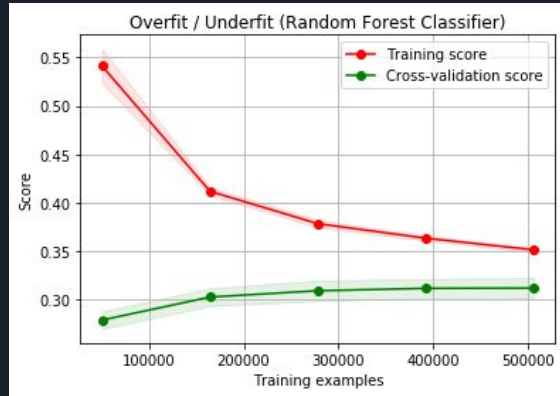
Component	RMSE Error (Test Set)
Overall System	1.6283
isMobile	1.6545
hits	1.6602
continent	1.7000
pageviews	1.9476



Drawbacks

- The dataset has features with many categorical values corresponding to every feature.
- Using one-hot encoding introduces a lot of features (computationally infeasible)

Random Forest - Learning curves and analysis



As can be seen from the above graphs, the model obtained is well fitted with train and test errors almost near each other.



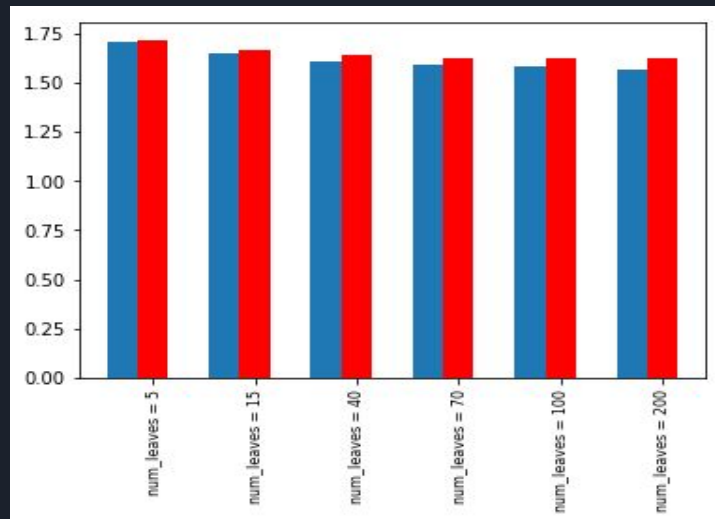
Why it performs better than LASSO?

- Bagging Method - Combination of different models (trees)
- Searches for the best feature from a randomly chosen subset of features while splitting a node
- Rather than taking all the features into consideration at each iteration, it considers the mse scores wrt one feature at a node split - decreasing the loss more efficiently.

Parameter Tuning

Best Hyperparameters Obtained:

- `max_leaf_nodes = 40`
- `max_depth = 8`
- `max_features = 10`





Issues with Random Forests

- It is difficult to get insight into how the decision function worked on the split for each node.
- If there is no check on the number of leaf nodes and max depth it may overfit.
- As we increase the number of estimators computation becomes heavy.



Feature Importances - Random Forests

<i>Feature name</i>	<i>Feature Importance</i>
Date	0.0081
Full Visitor ID	0.0009
Session ID	0.0008
Visit ID	0.0156
Visit Number	0.0406
Visit Start Time	0.0168
isMobile	0.0079
bounces	0.0053




Feature Importances

<i>Feature name</i>	<i>Feature Importance</i>
hits	0.2535
newVisits	0.0360
Page Views	0.4205
Visits	0
Browser type	0.0008
Operating System	0.0067
Device Category	0.0078
Network Domain	0.0008



Feature Importances

<i>Feature name</i>	<i>Feature Importance</i>
Channel Grouping	0.0075
Continent	0.0337
City	0.0030
Country	0.0923
Region	0.0042
SubContinent	0.0072
Source	0.0290



Feature Importance with PCA to reduce to 10 features

['date', 'visitNumber', 'isMobile', 'bounces', 'hits', 'newVisits', 'pageviews', 'visits', 'browser_cat', 'operatingSystem_cat', 'deviceCategory_cat', 'networkDomain_cat', 'channelGrouping_cat', 'continent_cat', 'city_cat', 'country_cat', 'region_cat', 'subContinent_cat', 'source_cat']

[0.06754012, 0.01422382, 0.01348026, 0.06783901, 0.01488425, 0.12380396,
0.52627007, 0.09597222, 0.05589025, 0.02009604]

RMSE is 1.682793374991196