COL341
Report A1.2

Ishaan Watts
2019PH10629

Part-C

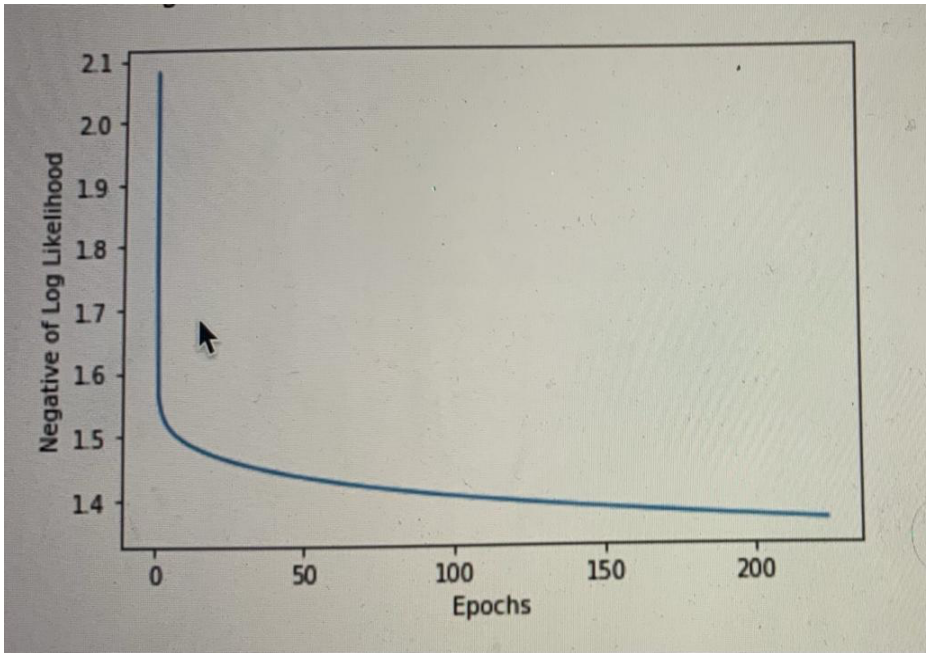
Performed hyper-parameter tuning with train.csv and test.csv and using cross-entropy-loss as the criterion. All possible combinations were run for 10mins on Google-Colab.

Varied parameters:
1. Type of Learning Rate
2. Batch Size
3. Learning Rate

A) Constant Learning Rate

Learning Rates = [0.01, 0.03, 0.1, 0.3]
Batch Sizes = [1, 4, 16, 64, 128, 256, 512]

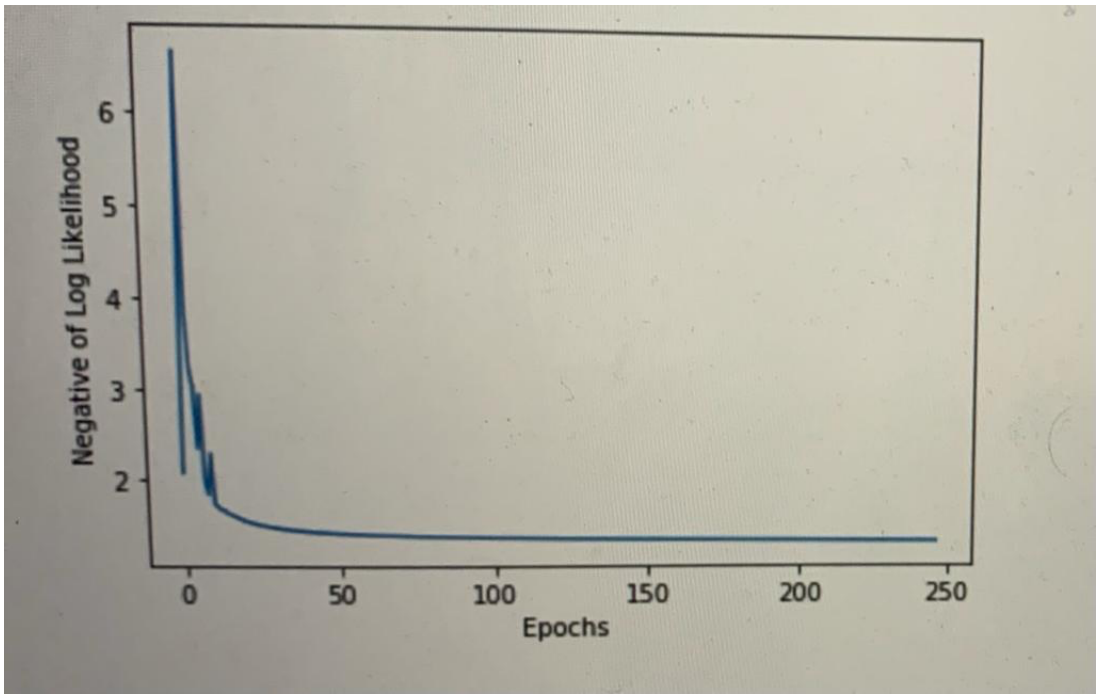
Best Output:
Learning Rate --> 0.1, Batch Size --> 16, Loss --> 1.3632981508140367



B) Adaptive Learning Rate

Learning Rates = [0.1, 0.3, 1, 3, 10, 30]
Batch Sizes = [1, 4, 16, 64, 128, 256, 512]

Best Output:
Learning Rate --> 10, Batch Size --> 64, Loss --> 1.3110783484218458



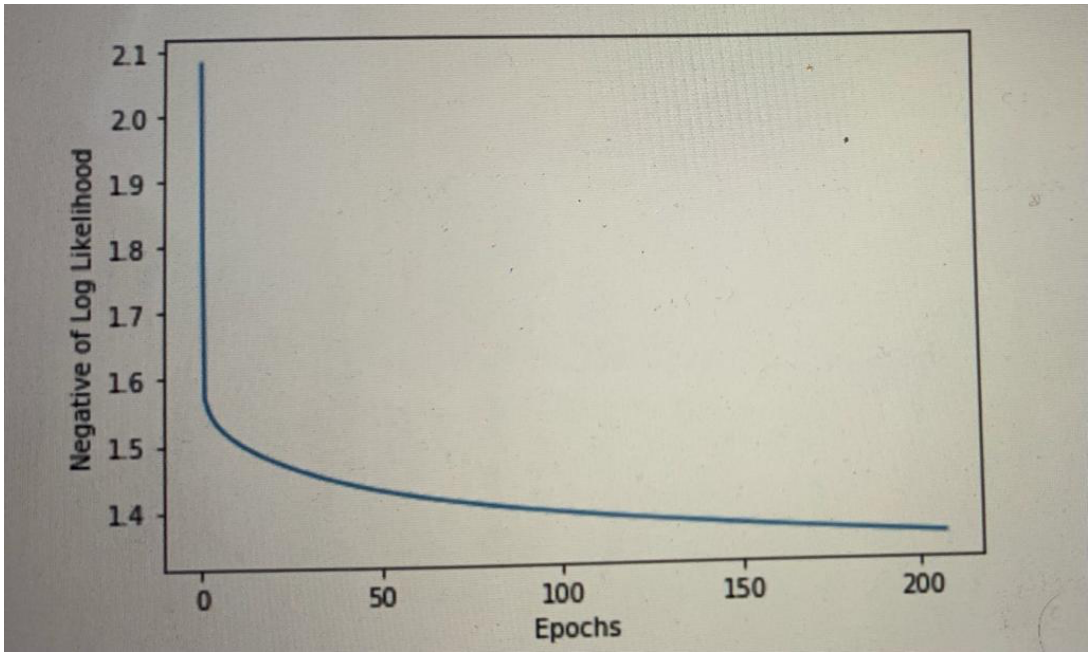
These set of parameters were giving NaN on train_large.csv as obvious with the peak at the start, so the next best non-NaN parameters were chosen.

Learning Rate --> 1, Batch Size --> 64, Loss --> 1.3766901745203867
(On train_large.csv)

C) Alpha-Beta Backtracking Learning Rate

Learning Rates = [0.03, 0.1, 0.3, 1, 3]
Batch Sizes = [1, 4, 16, 64, 128, 256, 512]
Alpha = 0.5, Beta = 0.9

Best Output:
Learning Rate --> 1, Batch Size --> 64, Loss --> 1.3510939410868752



Final Parameters Chosen:

Many of the parameters were giving NaN values on train_large.csv

Adaptive Learning Rate
Learning Rate --> 1
Batch Size --> 64

param_search.py

Line 1-90 --> Loading and preprocessing data, defining functions used in part-b
Line 92-95 --> Parameters to be varied

Line 97-130 --> Applying parameter search using for loops
Line 117-121 --> Plotting graphs for each combination (commented)

Part-D

Performed Feature Engineering on train.csv and the performed hyperparameter tuning as done in part-c.

Imported Logistic Regression from sklearn.linear_model as a benchmark to check scores on different features created.

Used SelectKBest from sklearn.feature_selection to select top 500 features.
Score functions used:
1. chi2
2. f_classif (ANNOVA based feature selection)

Methods tried:

1. Used run and encode to one-hot encode features excluding Total Costs. Selected top500 features first using chi2 --> **43.664**, f_classif --> **43.57**

2. Created polynomial features in addition to method 1. Combined different combinations of polynomial and one-hot features.
400 one-hot and 100 polynomial --> **22.67**
450 one-hot and 50 polynomial --> **32.56**

3. Encoding features with less than 51 unique values as one-hot and created polynomial features for remaining features. Score --> 22.13

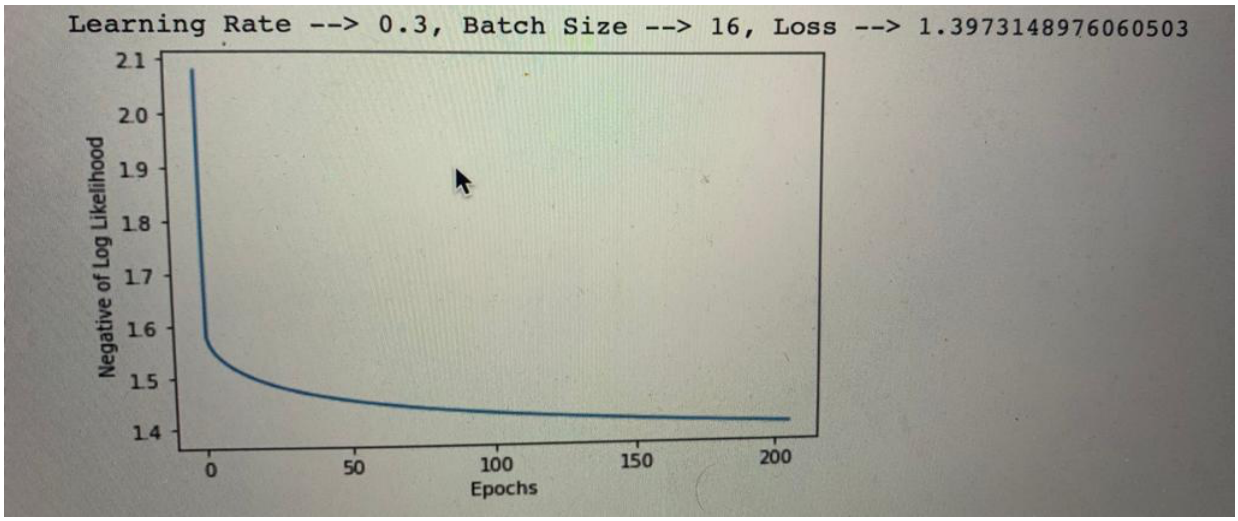
Final Features:

I concluded polynomial features were decreasing my score and hence used top500 one-hot features using chi2 as the score function.

Hyper-Parameter Tuning performed similarly as Part-C

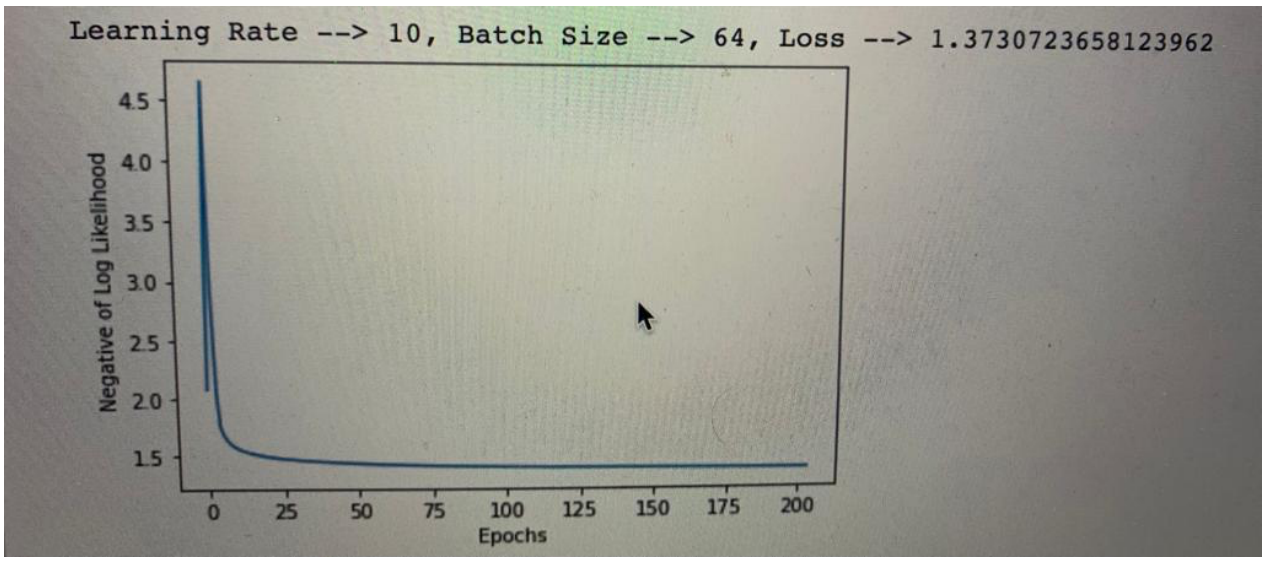
A) Constant

Best Output:
Learning Rate --> 0.3, Batch Size --> 16, Loss --> 1.397314896060503



B) Adaptive

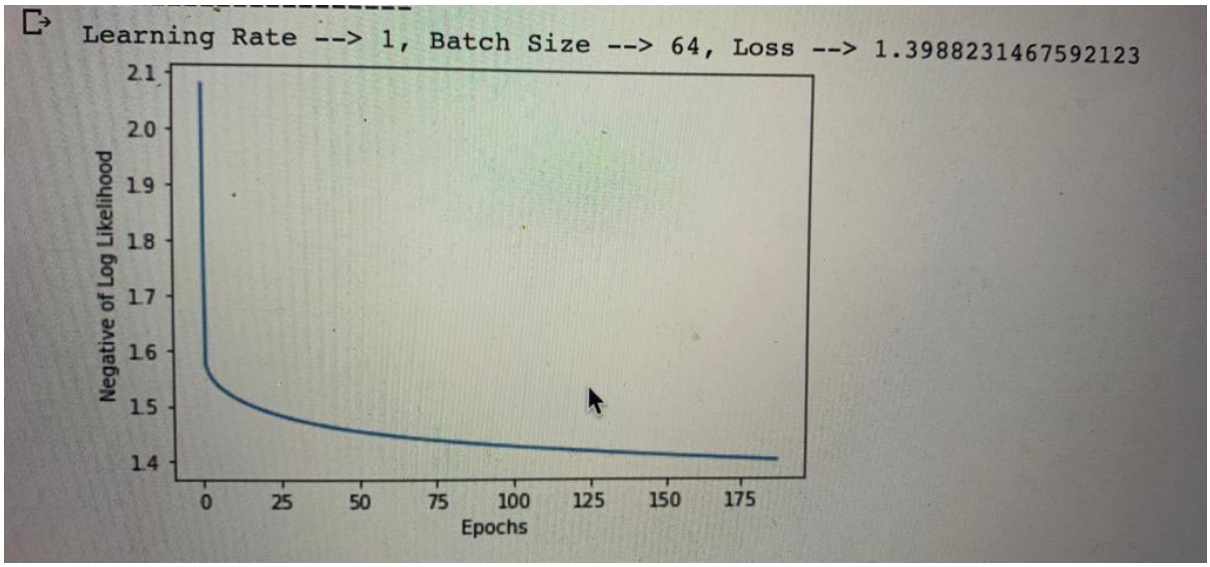
Best Output:
Learning Rate --> 10, Batch Size --> 64, Loss --> 1.3730723658123962



Again, it is exploding on train_large.csv as evident from the peak.
Choosing next best non-NaN value
Learning Rate --> 1, Batch Size --> 64, Loss --> 1.3889472839652110
(On train_large.csv)

C) Alpha-Beta Backtrack

Best Output:
Learning Rate --> 1, Batch Size --> 64, Loss --> 1.3988231467592123



Final Parameters Chosen:

Many of the parameters were giving NaN values on train_large.csv

Adaptive Learning Rate
Learning Rate --> 1
Batch Size --> 64

logistic_features_selection.py

Line 1-97 --> Loading and preprocessing data, defining functions used in part-b
    Line 40-42 --> Top 500 feature selection using SelectKBest with chi2 scorer

Line 100-103 --> Parameters to be varied

Line 105-146 --> Applying parameter search using for loops
    Line 126-129 --> Plotting graphs for each combination (commented)