Section 4.2

(1) Drop columns with majority rows as null and columns with more than 50% uniqueness

As observed in EDA, these columns have many missing values and therefore, will not be as useful as other columns in the model: 1, 10, 12, 32, 33, 35, 38, 39.

Also, these categorical columns have high uniqueness in the number of categories as explained in Section 2.6 and will thus be dropped from the dataset so that the approach is scalable: 16, 17, 20, 23, 24, 25, 26, 28, 29, 31, 34, 37.

(3) Run log(x+1) transformations on numeric variables

A log transformation is applied at this step to remove the high skewness in the data observed in EDA. Instead of log(x), log(x+1) is used to handle datapoints that have a value of 0.

(4) Fill null with mean for numeric variables

Data points with missing values are filled with mean instead of 0 to avoid skewing the data. The row is not ignored because it may contain useful features in other columns.

(5) Fill null with zero for categorical variables

Since categorical variables do not have a mean, they will be filled with 0, which becomes another category.

(6) Perform one-hot encoding for categorical variables

Next, we one-hot encode the categorical variables so that the categories become a form which can be used by the regression model. We make use of StringIndexer from the pyspark ml library which indicates the category index of that datapoint. For example, the value of 2.0 in the first row of column 36\_indexed tells us that the datapoint corresponds to the second category in the list of categories in column 36.

Then, OneHotEstimater one-hot encodes each specified column in a sparse way. For the same example, the first row of 36\_indexed\_encoded yields (11, [2], [1.0]) which means that there are 11 categories in column 36, the datapoint belongs to second category.

Finally, VectorAssembler is used to collect all the encoded features into a vector, 'CatFeatures' in our case. (8592, [0, 277, 1090...]) would mean that there is a total of 8592 features expanded from encoding the categorical features and the first row is 1.0 for category 0, 277, 1090 etc.

(7) Perfom standard normalization for categorical variables

We then make use of pyspark MLlib's StandardScaler is applied to 'CatFeatures' to normalize each feature to have unit standard deviation and/or zero mean.

(8) Perform normalization for numeric variables

Following that, the log-transformed, imputed numeric columns are assembled into another vector and normalized.

(9) Finalize data transformation

The transformed and normalized numeric and categorical features are now assembled together into one feature vector.

(10) PCA

As we expect the total number of features will not scale well together for such a big dataset, Principal Component Analysis (PCA) will be used to reduce to 100 features.

(11) Finalize dataframe

The pca\_features and target columns are extracted into a final dataframe, ready for model training.

**Section 4.3.**

**The 1000 point datset is split into 80-20 for training and development purpose. This set would help ensure that the codes are ready to be used on the full-size dataset.**

(1)

We made use of pyspark MLlib’s logistic regression to fit the toy dataset and predicted result with the development set.

(2) pred accuracy

We obtained an accuracy of 0.743 by computing the number of correct predictions over total predictions made.

(3)

We also computed log loss for the dev set.