CS6510 - APPLIED MACHINE LEARNING

Kaggle Challenge 2019

Saksham Mittal

14.04.2019
CS16BTECH11032

Om Sitapara

14.04.2019
CS16BTECH11036

Data Preprocessing:

Here we have used pandas for preprocessing of data. First, we removed the **id** column from the training set. Then we made a separate array called **training_labels** for the **pricing_category** and removed the **pricing_category** from the **training_set**. For the attributes with **nan**, we did the following processing:

(these changes are the same for both test and train data)

1) **Taxi_type**: We filled the nan with 'O' defining a new class type. Then we used one hot encoding to encode the labels. (We also tried the label encoding but it performed poorly)
2) **Customer_score**: We filled the nan for this category with the mean of all customer_score.
3) **Customer_score_confidence**: We filled the nan with 'O' defining a new class and then we used one hot encoding for the labels.
4) **Months_of_activity**: We filled the **nan** with 0.0
5) **Sex**: We used one hot encoding for male and female.
6) **Drop_location_type**: We used one hot encoding for different label types.
7) **Anon_var_1**: Here we tried two things:
    - We replaced the **nan** with the mean of this feature
    - Drop this column since the number of **nan** values were high.

8) We also tried to normalize the data but it was performing poor than non-normalized data.

Model :

For the main model, we have used **XGBoost** (Extreme gradient boost). The parameters for the classifier are:

1) Objective = ‘multi:softmax’: Instead of default reg:linear we used multi:softmax because this objective performs better for multiclass classification using softmax.
2) Num_class = 3: This attribute is required when the objective is set as multi:softmax for defining the number of unique classes.
3) Colsample_bytree = 0.8: This is a fraction of columns to be randomly sampled for each tree.
4) Subsample = 0.8: This is the fraction of observations to be randomly sampled for each tree. (We stared from 1 and reduced it to 0.8 to prevent overfitting)
5) Scale_pos_weight = 1 : This attribute signifies high class imbalance.(We noticed class 2 was more frequent than other classes)
6) Learning_rate = 0.06: We started the learning rate to be 0.1 and then slowly reduced till 0.04, and found 0.06 to be best.
7) Max_depth = 5: This attribute shows the max depth of the trees. Used to control overfitting. We reduced this from 6 to 5.
8) N_estimators = 500: This is the number of trees (weak learners) to be made.
9) Gamma = 5: This specifies the minimum loss reduction required to make a split. This made algorithm conservative.

Other Models we Tried:

- **Random forests**: We tried random forests with fine-tuning its parameters. This led to reach max accuracy of 0.6930
- **Logistic regression**: We tried logistic regression which led to an accuracy of 0.692
- **SVM**: We tried SVM with gamma as scale and decision_function_shape = ‘ovo’ for multiclass classification (1vs all). This led to an accuracy of 0.695
- **Neural networks**: We tried two different models for neural nets:
    1) Using Keras we defined our neural net as h1 : 12, h2 : 8, h3 : 5 and final output as 3 and trained on learning rate = 0.1 with number of epoch = 15 with batch_size = 10 leading to accuracy of 0.652

    2) For the second neural net, we imported the MLP class from the Sklearn where we kept the gradient = ‘adaptive’ leading to an accuracy of 0.684

- **Adaboost**: We tried Sklearn adaboost with fine tuning some parameters leading to an accuracy of ~0.68
- **Ensembled Model**: Here we tried best of 5 (Random forest, SVM, Neural net, adaboost, Logistic regression) which led to an accuracy of 0.699