Churn Prediction

Yongrae Jo

Abstract

This report contains a churn prediction conducted during term project execution results of the Big Data Processing held at fall, 2020 at POSTECH.

1 Project Overview and Problem Definition

This term project aims to build a model and conduct evaluation to predict whether a user will not become a chuner after 30 days after signing up based on given user data.

The definition of Chunrer means that the user leaves the group he belongs to, and specifically, it is defined as a user who has not posted any posts for 180 days after 30 days of account creation. In contrast, a Stayer is defined as a user who has posted at least one post for 180 days from 30 days after account creation.

The data provided are Users table and Posts table, and the Users table is numbered with a flag indicating user ID, account creation date, and whether or not to churn. The Posts table shows the information of many posted posts, and includes information such as PostID, PostTypeId, AcceptedAnswerId, CreationDate, Answer Count, Score, Comment Count, Parent Id, BodyWordNum, OwnerUserId, etc.

The purpose of this project is to predict the churn of new users by analyzing the post posting pattern and churn of each user based on the Users table and Posts table above. For this, it is necessary to construct a predictive model using machine learning techniques based on the given user table data. Also, considering the size of the Posts table, it is necessary to use a suitable tool to handle big data. In order to objectively evaluate the constructed prediction model, we measure the prediction accuracy of the model using two metrics, F1-Score and AUC, using validation data provided by the assistant.

The project proceeds in the order of data loading, feature extraction, model design, model train, and evaluation, referring to the existing ML-pipeline model.

Spark v3.0.1 was used as the data analysis platform used in the project, and sparkcluser was deployed as a standalone and implemented on the local machine. I used pyspark and Dataframe for programming interfaces, and ml for machine learning libraries.

2 Data Load

The data is stored in the local file system, and posts_dist.csv, users_train_dist.csv, users_val_dist.csv are loaded as Resilient Data Type (RDD) using spark's textFile method. Since the recent RDD is going to be deprecated in maintainence mode, it was converted into Dataframe and saved. For conversion, the function convertToDataFrameFromRDD is defined.

3 Feature Extraction and Preprocessing

3.1 Defining feature

A first important feature that determines whether to churn is the number of posts within a specific period. Therefore, in order to obtain a feature for this, a feature representing the time from the creation of a user account to the time of a post and the post posted by the user in days was defined. This feature can be expressed as a categorical variable that has three major values, and it is defined as "Early" if it is 30 or less, "Mid" if it is less than 210 days, and "Late" otherwise. To explain intuitively the reason why this is important, I believe that the number of posts in a specific period (especially early days after joining the group) shows that the person has a passion to participate in discussing the technicalities in the forum. And this feature can represent

Id	Questions	Answer	Early	Mid	Late	sum_scr	avg_scr	std_scr	sum_ans	avg_ans	std_ans	sum_cmt	avg_cmt	std_cmt	sum_wrd	avg_wrd	std_wrd	AcceptedAnswerCnt	isChurn
148	4	49	0	40	13	1037	19.5660	67.7483	20	0.3774	1.547	96	1.8113	2.2279	5362	101.1698	62.6801	16	false
833	0	1	0	1	0	0	0.0000	0.0	0	0.0000	0.0	0	0.0000	0.0	41	41.0000	0.0	0	false
1088	12	315	16	65	246	3230	9.8777	69.2961	50	0.1529	1.2364	425	1.2997	2.1455	30137	92.1621	82.2279	38	false
1580	20	44	12	7	45	259	4.0469	13.8007	67	1.0469	2.0581	40	0.6250	1.1198	5350	83.5938	69.3358	8	false
1591	0	192	0	12	180	504	2.6250	5.5706	0	0.0000	0.0	272	1.4167	1.8229	8316	43.3125	40.9026	30	false
1645	1	4	1	3	1	87	17.4000	18.2839	12	2.4000	5.3666	3	0.6000	0.8944	369	73.8000	59.3734	0	false
1959	18	14	4	9	19	785	24.5313	73.4579	85	2.6563	3.3371	49	1.5313	2.1249	3946	123.3125	98.9475	2	false
2122	37	14	3	11	37	828	16.2353	43.1632	90	1.7647	1.7617	51	1.0000	1.7321	16463	322.8039	198.1531	6	false
3175	20	9	5	10	14	397	13.6897	41.2182	136	4.6897	9.3009	39	1.3448	2.1921	3876	133.6552	112.5053	3	false
3749	1	1	2	0	0	8	4.0000	1.4142	4	2.0000	2.8284	0	0.0000	0.0	77	38.5000	19.0919	1	true
4101	4	1	4	0	1	27	5.4000	5.5946	15	3.0000	2.2361	0	0.0000	0.0	438	87.6000	30.509	0	true
5156	2	19	11	0	10	1487	70.8095	195.52	22	1.0476	4.3644	41	1.9524	3.7746	1377	65.5714	47.2203	6	true
6336	4	31	23	8	4	140	4.0000	9.6954	26	0.7429	2.3432	33	0.9429	1.3272	3449	98.5429	56.4568	3	false
8638	20	5	2	1	22	285	11.4000	14.428	76	3.0400	2.8059	16	0.6400	1.0755	4082	163.2800	115.805	3	false
9465	4	299	1	6	296	1231	4.0627	12.5006	7	0.0231	0.2217	525	1.7327	2.0143	19701	65.0198	58.1986	91	false
9852	0	7	5	2	0	23	3.2857	3.0394	0	0.0000	0.0	7	1.0000	0.8165	999	142.7143	55.8472	0	false
9900	3	13	3	0	13	690	43.1250	138.7928	15	0.9375	3.2346	24	1.5000	3.0984	1626	101.6250	79.474	4	true
10817	1	0	0	0	1	14	14.0000	0.0	7	7.0000	0.0	3	3.0000	0.0	154	154.0000	0.0	0	true
11858	13	140	11	3	139	5772	37.7255	375.8966	45	0.2941	1.0875	215	1.4052	2.2608	8725	57.0261	47.1209	21	false
12799	6	30	1	2	33	259	7.1944	13.9567	15	0.4167	1.079	55	1.5278	2.3843	3080	85.5556	68.5071	2	false

Figure 1: Per-User Statistics

the basis for determining whether or not the person will remain in the forum.

Second important features include the number of question/answer posts for each user, and the number of accepted posts in case of answer posts. Especially, the number of accepted answers post can be the most significant impact on the user because making those answers usually needs a non-trivial effort, and accepted answers posts serves a kind of incentive mechanism in the forum by contributing the body of knowledge.

Third features includes numerical measurements that can be directly induced from the table. Features such as score, number of received answer posts, comment counts, word counts are given in the train data. From this raw, I draw an average, summation, and standard deviation.

Overall, I defined the following features to construct a per-user statistics.

- Question Posts: The number of question post (type 1) that the user had posted
- Answer Posts: The number of answer posts (type 2) that the user had posted
- Accepted Answer count: The number of accepted posts that the user had posted
- Early: The number of posts within the first 30 days.
- Mid: The number of posts between the 31th days and 210th days.
- Late: The number of posts after 210th day
- SumAnswers, AvgAnswers, StdAnswers: The sum, avg, std of the number received answer post of type 1 post.
- SumScore, AvgScore, StdScore: The sum, avg, std of the number received score.
- SumComment, AvgComment, StdComment: The sum, avg, std of the number received comment.
- SumWords, AvgWords, StdWords: The sum, avg, std of the number of words in each post.
- Id: A unique user identity.
- IsChurn: Represents the user with Id is churner or not.

In order to extract those features from the given data, it is necessary to match the user ID and each post registered user ID, so a join operation between the two files, posts_dist.csv and users_train_dist.csv, is required.

After the Join operation above, per-user total statistics were calculated by grouping each post into a group. (Fig 3) Here, the total number of posts for each user, the total score sum, and the total answer It includes the number, total number of comments, total number of words, and the sum of the number of posts per period (Early, Mid, Late). The reason why the total sum for each user is selected as a feature is that it may not be a delicate feature, but roughly, it seems that the total sum can indicate whether the user churn or not.

3.2 Preprocessing

As in the fig 3, the features have different scale. The those scale difference can have a bias towards to larger feature, which possibly results in a wrong prediction model. To resolve this, we need to normalize each columns in those features. Specifically, the features that is needed to learn machine learning model, are all features but Id, *isChurn*. The *isChurn* is exempt because it is label column.

Questions	Answer	Early	Mid	Late	sum_scr	avg_scr	std_scr	sum_ans	avg_ans	std_ans	sum_cmt	avg_cmt	std_cmt	sum_wrd	avg_wrd	std_wrd	AcceptedAnswerCnt
0.232	0.615	0.0	1.889	0.186	1.442	0.705	2.47	0.427	0.261	1.498	0.609	1.305	1.942	0.651	1.032	0.929	0.774
0.0	0.013	0.0	0.047	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.418	0.0	0.0
0.697	3.955	2.122	3.07	3.517	4.492	0.356	2.527	1.069	0.106	1.197	2.696	0.937	1.87	3.657	0.94	1.219	1.839
1.162	0.552	1.592	0.331	0.643	0.36	0.146	0.503	1.432	0.723	1.993	0.254	0.45	0.976	0.649	0.853	1.028	0.387
0.0	2.411	0.0	0.567	2.573	0.701	0.095	0.203	0.0	0.0	0.0	1.726	1.021	1.589	1.009	0.442	0.606	1.452
0.058	0.05	0.133	0.142	0.014	0.121	0.627	0.667	0.256	1.658	5.196	0.019	0.432	0.78	0.045	0.753	0.88	0.0
1.046	0.176	0.531	0.425	0.272	1.092	0.884	2.679	1.817	1.835	3.231	0.311	1.104	1.852	0.479	1.258	1.467	0.097
2.149	0.176	0.398	0.52	0.529	1.152	0.585	1.574	1.923	1.219	1.706	0.324	0.721	1.51	1.998	3.294	2.938	0.29
1.162	0.113	0.663	0.472	0.2	0.552	0.493	1.503	2.906	3.24	9.004	0.247	0.969	1.911	0.47	1.364	1.668	0.145
0.058	0.013	0.265	0.0	0.0	0.011	0.144	0.052	0.085	1.382	2.738	0.0	0.0	0.0	0.009	0.393	0.283	0.048
0.232	0.013	0.531	0.0	0.014	0.038	0.195	0.204	0.321	2.073	2.165	0.0	0.0	0.0	0.053	0.894	0.452	0.0
0.116	0.239	1.459	0.0	0.143	2.068	2.552	7.129	0.47	0.724	4.225	0.26	1.407	3.29	0.167	0.669	0.7	0.29
0.232	0.389	3.051	0.378	0.057	0.195	0.144	0.354	0.556	0.513	2.269	0.209	0.68	1.157	0.419	1.006	0.837	0.145
1.162	0.063	0.265	0.047	0.314	0.396	0.411	0.526	1.624	2.101	2.716	0.102	0.461	0.937	0.495	1.666	1.717	0.145
0.232	3.754	0.133	0.283	4.231	1.712	0.146	0.456	0.15	0.016	0.215	3.331	1.249	1.756	2.391	0.664	0.863	4.405
0.0	0.088	0.663	0.094	0.0	0.032	0.118	0.111	0.0	0.0	0.0	0.044	0.721	0.712	0.121	1.456	0.828	0.0
0.174	0.163	0.398	0.0	0.186	0.96	1.554	5.061	0.321	0.648	3.132	0.152	1.081	2.7	0.197	1.037	1.178	0.194
0.058	0.0	0.0	0.0	0.014	0.019	0.504	0.0	0.15	4.837	0.0	0.019	2.162	0.0	0.019	1.572	0.0	0.0
0.755	1.758	1.459	0.142	1.987	8.028	1.359	13.707	0.962	0.203	1.053	1.364	1.013	1.97	1.059	0.582	0.699	1.017
0.349	0.377	0.133	0.094	0.472	0.36	0.259	0.509	0.321	0.288	1.045	0.349	1.101	2.078	0.374	0.873	1.016	0.097

Figure 2: Per-user statistics (normalized)

```python
from pyspark.ml.feature import VectorAssembler
from pyspark.mllib.classification import SVMWithSGD, SVMModel
from pyspark.sql.functions import col
from pyspark.mllib.regression import LabeledPoint
from pyspark.ml.classification import LinearSVC

assembler = VectorAssembler(
      inputCols=["TotalPost", "TotalScore", "TotalAnswers", "TotalCmt", "TotalWords", "#Early", "#Mid", "#Late"],
      outputCol="features")

train_df_transformed = assembler.transform(train_df)
final_df = train_df_transformed.select(col("isChurn").alias("labellabel"), col("features"))

lsvc = LinearSVC(maxIter=10, regParam=0.1)
lsvcModel = lsvc.fit(final_df)
```

Attempting to reconnect...

Figure 3: Model selection and fitting

We used StandardScaler, which normalizes each feature to have unit standard deviation and/or zero mean. The formula have the following expressions.

$$z = \frac{(x_i - u)}{s}$$

where z is normalized value, x is the each value of each feature vector, u is the mean of each feature, and s is the standard deviation of the feature.

As a result, we have the normalized feature table as in fig 2.

3.3 Applying dimension reduction technique?

Before moving onto the model selection, we need to check whether the dimensions of the feature can whether be reduced or not. Before applying dimension reduction techniques such as Principal Component Analysis (PCA), Singular Value Decomposition, Random Projection, we need to see the correlation relationship between features because the measurement represents similarities, which allows to multiple features can be grouped into a single feature with minimizing the variance of original data. We plot the correlation matrix as a heatmap as in fig 4. In the figure, high correlation measurements are shown with more brighter colors. And low correlation measurements are shown with more darker colors. As a result, there are some similarities between features, for example, a pair of Answer and Late. However, we decides not to take any dimension reduction techniques because those correlation does not statistically significant meaningful. Note that the data used in the plot is randomly sampled due to large size of original train dataset.

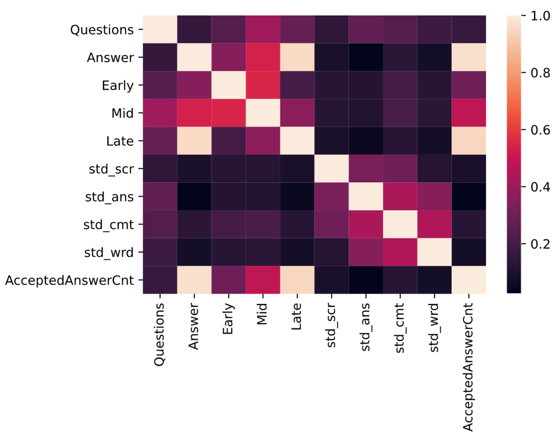

Figure 4: Features Correlation matrix as a heatmap

```
def ConvertToFeatureLabel(df, feature_cols):
    df = df.withColumn("isChurn" , F.when( F.col("isChurn")=="True" , F.lit(1) ).otherwise(0) )
    df = df.withColumnRenamed("isChurn", "label")
    assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
    df = assembler.transform(df)
    return df

train_df = ConvertToFeatureLabel(train_df, features)
test_df = ConvertToFeatureLabel(test_df, features)
val_df = ConvertToFeatureLabel(val_df, features)
```

Figure 5: Converting a dataframe to the one having features and labels

```
from pyspark.ml.classification import LinearSVC

def Train(df, feature_cols):
    lsvc = LinearSVC(maxIter=10, regParam=0.1)
    model = lsvc.fit(df)
    return model

train_model = Train(train_df, features)
```

Figure 6: Model fitting source code

4 Model Selection

We decide to take the supervised learning-based model because *isChurn* serves as the label column for each record in the given data. And since the *isChurn* is a binary variable which can only take *True* or *False*, we decided to use binary classification model.

With the above reasoning, we choose to use Support Vector Classification (SVC). LinearSVC, provided in pyspark's ml.classification library, is a binary classifier that optimizes the Hinge Loss using the OWLQN optimizer. It only supports L2 regularization currently (pyspark, 2020). Parameters of LinearSVC have been set to maxIter=10, regParam=0.1 for fitting the model. An implementation of model fitting with SVC is given in fig5 and fig 6. Note that in pyspark, each datarame should be converted to the one that have features and label column to use mahcine learning library. In our case, a feature column are assembled from the defined features above, and label column comes from isChurn column. After converting, fitting model with training data is performed.

5 Evaluation

5.1 Performance Evaluation

For evaluating the model, we used BinaryClassificationEvaluator and MulticlassClassificationEvaluator provided by ml.evaluation. This libraries support measurement of various metrics,including F1-score and AUC. With those two metrics, we run against test data and the following results are obtained: F1-score was calculated as 0.7165338008796209 and auc was calculated as 0.6706135136516691 as in fig 7.

6 Conclusion

In this paper, an experiment was conducted to construct a model that predicts user churn using a big data analysis platform, pyspark. The final result of the model is currently not satisfactory, but we hope to make a better solution by applying various data analysis techniques in the future.

References

pyspark. 2020. Pyspark's linearsvc.

```python
from pyspark.ml.evaluation import BinaryClassificationEvaluator, MulticlassClassificationEvaluator

# Create both evaluators
evaluatorMulti = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction")
evaluator = BinaryClassificationEvaluator(labelCol="label", rawPredictionCol="prediction", metricName='areaUnderROC')

predictionAndTarget = train_model.transform(test_df).select("label", "prediction")
f1 = evaluatorMulti.evaluate(predictionAndTarget, {evaluatorMulti.metricName: "f1"})
auc = evaluator.evaluate(predictionAndTarget)
```

```python
print(f1)
print(auc)
```

```
0.7165338008796209
0.6706135136516691
```

Figure 7: Evaluation