# Churn Prediction Using Machine Learning

Soumyadip Ghatak
A21032

# What does Churn mean in business?

Churn means customers or users who left the services or migrates to the competitor in the industry. It is very important for any organization to keep its existing customer and attract new ones if one of them fails it is bad for business. The goal is to explore the possibility of machine learning for churn prediction to retain a competitive edge in the industry.

# Why it is important to solve the problem?

In highly competitive markets, like gaming, hotel, subscription apps and casino chains operate in, being able to truly understand our customers is crucial to remaining competitive. To do this, we have to analyze the behavior of our customers and understand what are the attributes of people who may churn.

# Project Overview

**In this Project** we will analyse and build a churn model for a “company” called Sparkify. This company provides music streaming services to customers just like Spotify or Pandora.

To utilize the Spark, we need to set up a spark session in the beginning. We

can also do this without actually having a server by using the local cluster.

# About our dataset.

**There are two datasets that are available: a full dataset of 12GB, that requires the deployment of a cluster to parallelize computation, and a mini-dataset of 128MB. In this repository the focus is on the mini-dataset for now. But due to the nature of this data, we wrote all the code using the Spark Framework to allow easy refactoring on the code to have it run with clusters on a much larger dataset.**

# Step by step approach.

1. <u>Data Understanding</u>

   Included finding missing or random values, understanding the structure of the dataset, understanding column datatypes, finding useful features.

2. <u>Visualization</u>

   Everything that can be learned from visualization. This will help to visually understand which features are more suitable for solving the problem.

## 3. Feature Engineering

**At this step, we will select features based on the previous steps and combine them into one data frame.**

## 4.Modeling

**We will use four machine learning algorithms: Logistic Regression, Random Forest, Decision tree. Then we'll compare them based on metrics: Accuracy and F1.**

## 5.Model Evaluation and Validation

**At this step, we provide some discussion about the final parameters of the model.**

For this project, we have to classify customers who have churned. Not only we will classify these customers but we will build a model that would make predictions.

The approach taken to this problem was to firstly define what churn is. This was defined by using the event of 'Cancellation Confirmation'. If the user cancelled their service, they were defined as churn.

After defining churn, some features were extracted such as gender and level. These features were used and a model was trained and it was used to make predictions.

# Snapshot of our dataset.

| artist | auth | firstName | gender | itemInSession | lastName | length | level | location | method | page | registration | sessionId | song | status | ts | userAgent | userId |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Martha Tilston | Logged In | Colin | M | 50 | Freeman | 277.89016 | paid | Bakersfield, CA | PUT | NextSong | 1538173362000 | 29 | Rockpools | 200 | 1538352117000 | Mozilla/5.0 (Wind... | 30 |
| Five Iron Frenzy | Logged In | Micah | M | 79 | Long | 236.09424 | free | Boston-Cambridge-... | PUT | NextSong | 1538331630000 | 8 | Canada | 200 | 1538352180000 | "Mozilla/5.0 (Win... | 9 |
| Adam Lambert | Logged In | Colin | M | 51 | Freeman | 282.8273 | paid | Bakersfield, CA | PUT | NextSong | 1538173362000 | 29 | Time For Miracles | 200 | 1538352394000 | Mozilla/5.0 (Wind... | 30 |
| Enigma | Logged In | Micah | M | 80 | Long | 262.71302 | free | Boston-Cambridge-... | PUT | NextSong | 1538331630000 | 8 | Knocking On Forbi... | 200 | 1538352416000 | "Mozilla/5.0 (Win... | 9 |
| Daft Punk | Logged In | Colin | M | 52 | Freeman | 223.60771 | paid | Bakersfield, CA | PUT | NextSong | 1538173362000 | 29 | Harder Better Fas... | 200 | 1538352676000 | Mozilla/5.0 (Wind... | 30 |

only showing top 5 rows

# The variables in our dataset.

```
|-- artist: string (nullable = true)
|-- auth: string (nullable = true)
|-- firstName: string (nullable = true)
|-- gender: string (nullable = true)
|-- itemInSession: long (nullable = true)
|-- lastName: string (nullable = true)
|-- length: double (nullable = true)
|-- level: string (nullable = true)
|-- location: string (nullable = true)
|-- method: string (nullable = true)
|-- page: string (nullable = true)
|-- registration: long (nullable = true)
|-- sessionId: long (nullable = true)
|-- song: string (nullable = true)
|-- status: long (nullable = true)
|-- ts: long (nullable = true)
|-- userAgent: string (nullable = true)
|-- userId: string (nullable = true)
```

**There were 1101 missing userId records in this dataset. For this, we removed the rows where these were missing.**

```
[ ] len(df.select(['userId','sessionId','artist','song']).where(df.userId == '').collect())

    1101
```

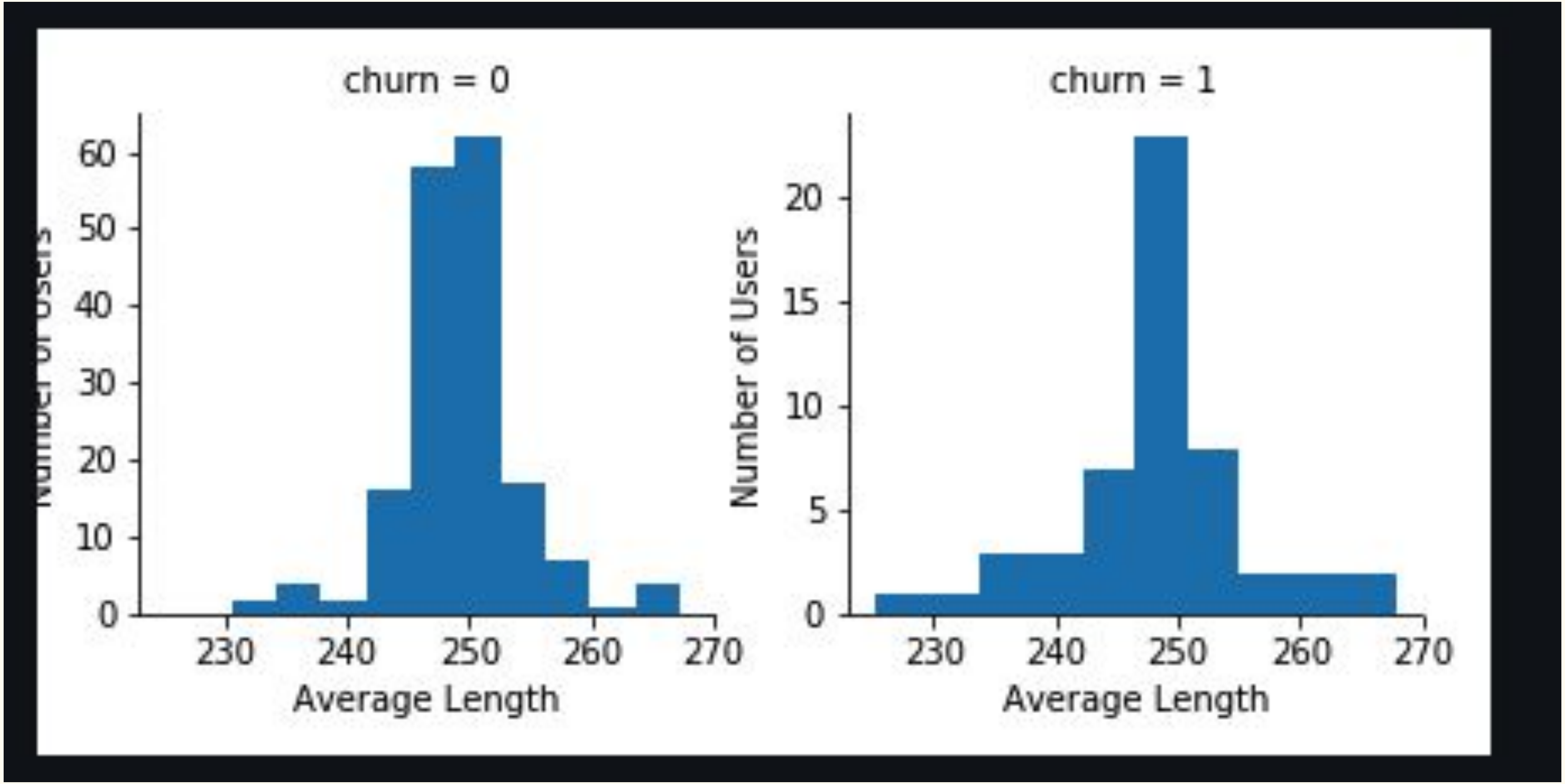
# We filter our dataset and extract the features that we will use.

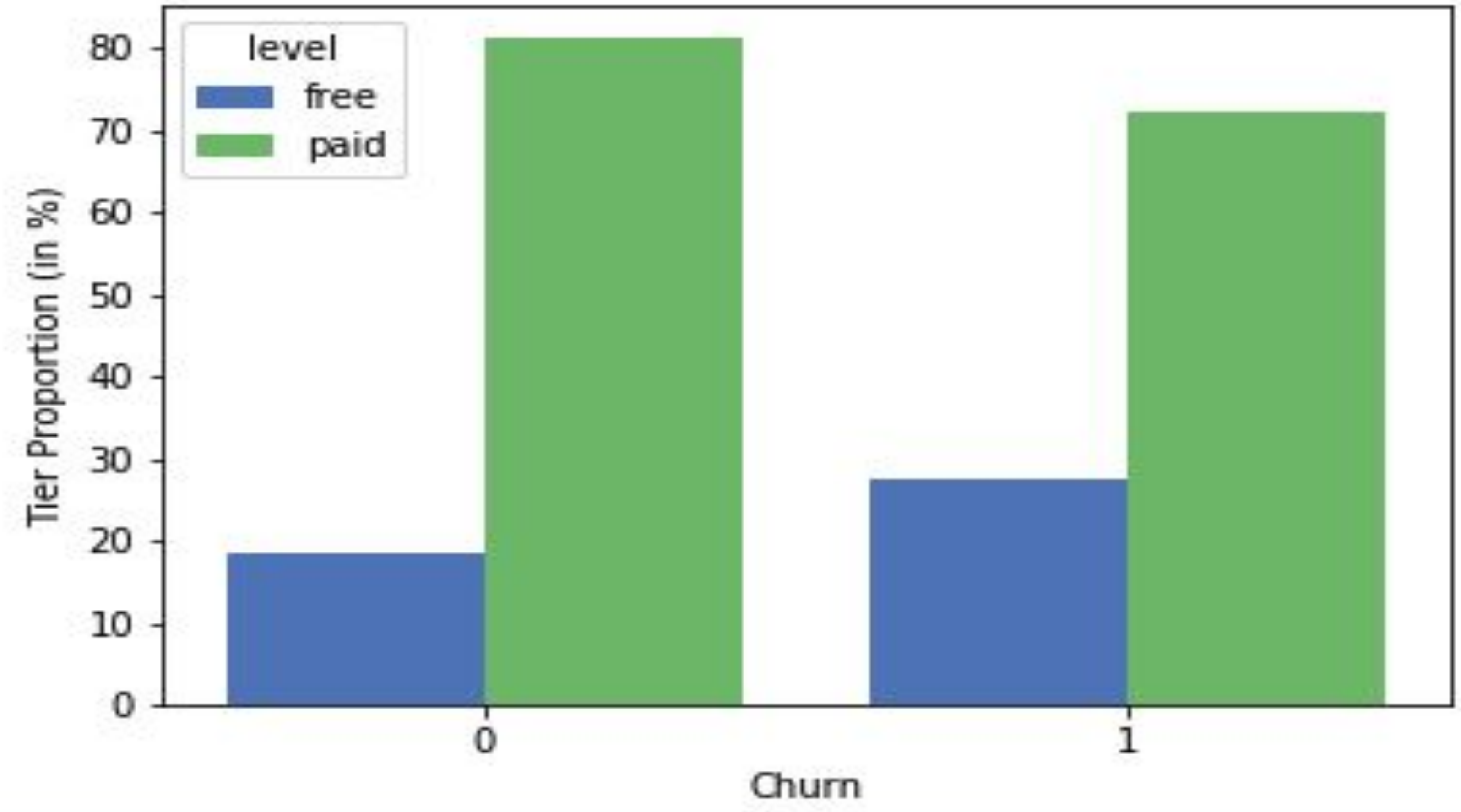
Based on intuition and domain knowledge, we decide not to include the columns for lastName, method, and userAgent in our first-pass modeling for now, since these variables probably do not affect our prediction. We also decide not to include artist, location, song, and status for now. This leaves us with the following columns

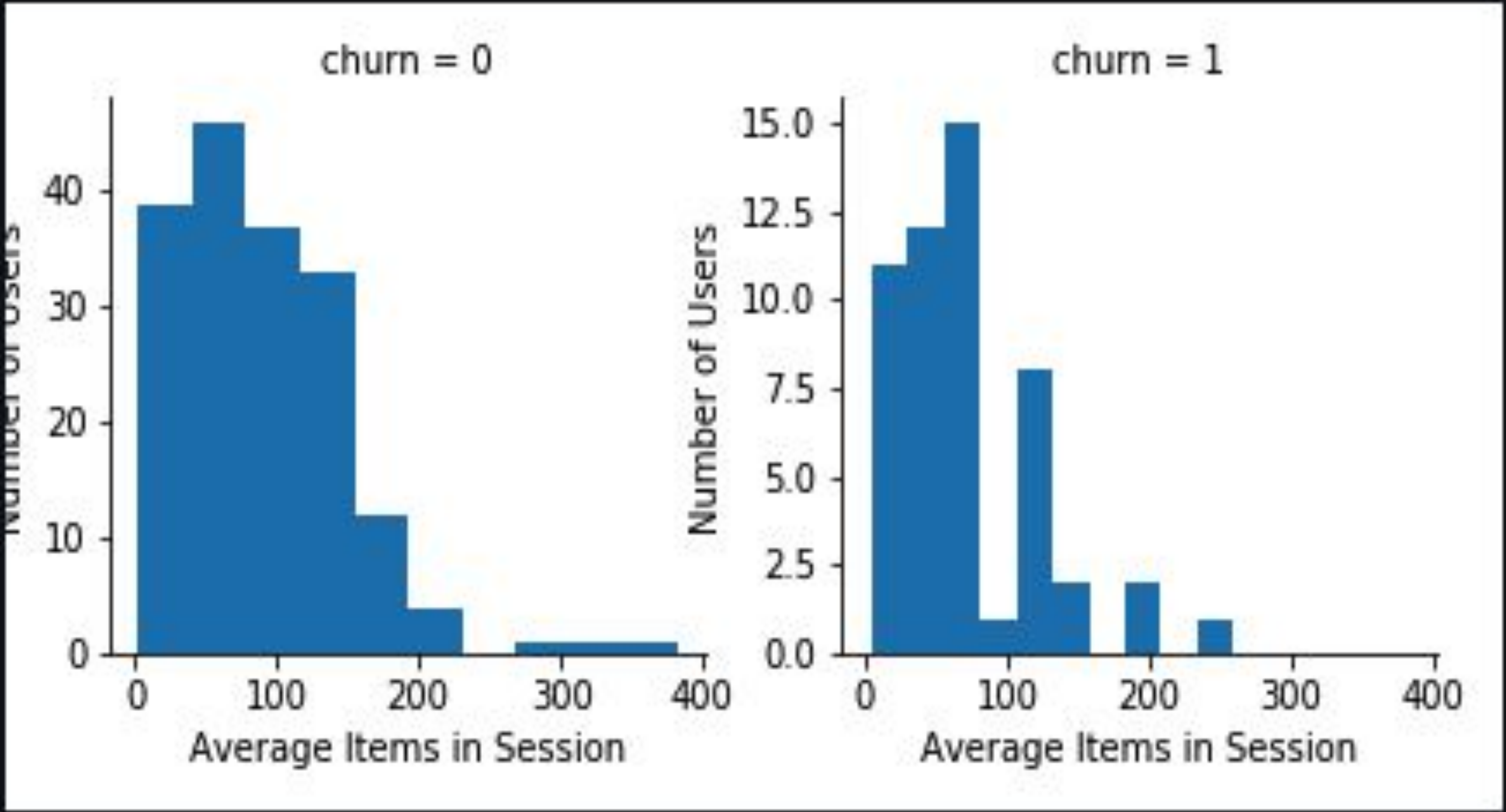
| artist | length | sessionId | datetime | firstname | itemInSession | song | userId |
|---|---|---|---|---|---|---|---|
| null | null | 141 | +50719-01-27 21:1... | Grant | 0 | null | 142 |
| The Kooks | 169.45587 | 141 | +50719-01-28 03:0... | Grant | 1 | Love It All | 142 |
| Plasmatics | 256.7571 | 141 | +50719-01-30 01:5... | Grant | 2 | The Damned | 142 |
| Alliance Ethnik | 195.94404 | 141 | +50719-02-02 01:0... | Grant | 3 | SinceritÃ Â© Et J... | 142 |
| Tears For Fears | 361.29914 | 141 | +50719-02-04 07:1... | Grant | 4 | Change | 142 |
| B5 | 189.36118 | 141 | +50719-02-08 11:3... | Grant | 5 | Heartbreak | 142 |
| null | null | 141 | +50719-02-08 18:1... | Grant | 6 | null | 142 |
| Plies | 278.22975 | 141 | +50719-02-10 16:0... | Grant | 7 | Somebody (Loves Y... | 142 |
| Yeasayer | 298.91873 | 141 | +50719-02-13 21:1... | Grant | 8 | Red Cave | 142 |
| null | null | 141 | +50719-02-14 06:0... | Grant | 9 | null | 142 |
| null | null | 141 | +50719-02-15 08:4... | Grant | 10 | null | 142 |
| null | null | 141 | +50719-02-15 09:0... | Grant | 11 | null | 142 |
| null | null | 141 | +50719-02-17 18:0... | Grant | 12 | null | 142 |
| Nickelback | 262.81751 | 141 | +50719-02-18 09:3... | Grant | 13 | I'd Come For You ... | 142 |
| Venetian Snares | 71.70567 | 141 | +50719-02-21 10:2... | Grant | 14 | Fuck Off | 142 |
| The Red Crayola | 153.80853 | 141 | +50719-02-22 06:0... | Grant | 15 | Transparent Radia... | 142 |
| Tune Up! vs. Ital... | 319.73832 | 141 | +50719-02-24 00:3... | Grant | 16 | Colours Of The Ra... | 142 |
| Enya | 104.56771 | 141 | +50719-02-27 17:1... | Grant | 17 | Deireadh An Tuath | 142 |
| Binary Star | 54.22975 | 141 | +50719-02-28 22:0... | Grant | 18 | The KGB (Intro) | 142 |
| Erasure | 246.46485 | 141 | +50719-03-01 13:0... | Grant | 19 | Fingers And Thumb... | 142 |

only showing top 20 rows

# Some important visualization to derive insights.




level
free
paid
Tier Proportion (in %)
0
10
20
30
40
50
60
70
80
0
1
Churn


churn = 0
churn = 1
Number of Users
Number of Users
Average Items in Session
Average Items in Session
0
10
20
30
40
0.0
2.5
5.0
7.5
10.0
12.5
15.0
0
100
200
300
400

# The most present states for people who churned.

```
+-----+-----+-----+
|churn|state|count|
+-----+-----+-----+
|    0|   CA|39158|
|    0|   PA|23708|
|    0|   TX|22200|
|    0|   NH|18637|
|    0|   FL|11427|
+-----+-----+-----+
only showing top 5 rows

+-----+-----+-----+
|churn|state|count|
+-----+-----+-----+
|    1|   CA| 7613|
|    1|   CO| 4317|
|    1|   MS| 3839|
|    1|   WA| 3526|
|    1|   OH| 3173|
+-----+-----+-----+
only showing top 5 rows
```

# ML Models used:

## Logistic Regression

```
[ ] predictions = train_model(df_balance)
```

```
[ ] scores(predictions)
```

```
Accuracy Score:  0.7867647058823529
F1 Score:  0.8066666666666668
```

# Random Forest Classifier

```
[ ] predictions_rf = train_model(df_balance, model='rf')
```

```
[ ] scores(predictions_rf)
```

```
Accuracy Score:  0.8007968127490039
F1 Score:  0.8201438848920863
```

# Decision Tree Classifier

```
predictions_dt = train_model(df_balance, model='dt')
scores(predictions_dt)
```

```
Accuracy Score:  0.7754237288135594
F1 Score:  0.792156862745098
```

Let's check our model scores

```
*****Model Scores******
Logistic Regression
Accuracy Score:  0.7877551020408163
F1 Score:  0.7936507936507937
None
Random Forest Classifier
Accuracy Score:  0.8007968127490039
F1 Score:  0.8201438848920863
None
Decision Tree Classifier
Accuracy Score:  0.7754237288135594
F1 Score:  0.7921568627450980
None
```

# We tried to improve our logistic regression model but it wasn’t. This can be solved by extracting more features.

```
preds.filter(preds.label == preds.prediction).count() / preds.count()
```

```
0.7466666666666667
```

```
[ ] scores(preds)
```

```
Accuracy Score:  0.7466666666666667
F1 Score:  0.774703557312253
```

# Thank you.