Telecom Churn Prediction: Exploring Impacts of Scale and Computing Environment

Pranay Bhandare, Jungho Park, Harman Singh Bath

BIA 678 Final Project Report

Prof. David Belanger

5/8/20

I. Project Purpose

For our final project, we have decided to work on a customer churn classification model for a telecom company where we will be able to predict whether or not a customer will churn or not - meaning that if the customer will leave or not. There are data points on customers that can be analyzed to strategize on how to retain them when they realize they are unhappy with a company's services, which I will go into more detail in the next section of this report. Businesses want to maximize their number of customers. To achieve this goal, it is equally important to attract new ones but also retain existing customers. Building up and keeping a loyal clientele can be challenging, especially when customers are free to choose from a variety of telecommunication providers and their diverse products/services. Apart from the model development for this classification churn problem, we will also explore and compare the performance of 3 different models amongst big data technologies such as AWS. The 3 models we will tune are Logistic Regression (LR), Decision Tree (DT) and Linear Support Vector Machine (linear SVM). Our emphasis for this project was to really analyze the impact of scaling using AWS clusters and comparing it to local machine execution.

II. Data Description

We found a dataset on Kaggle which contains information about approximately 6000 users and main features regarding their tenure with the company as displayed in the visualization:

```
|-- _c0: integer (nullable = true)
|-- customerID: string (nullable = true)
|-- gender: string (nullable = true)
|-- SeniorCitizen: integer (nullable = true)
|-- Partner: string (nullable = true)
|-- Dependents: string (nullable = true)
|-- tenure: integer (nullable = true)
|-- PhoneService: string (nullable = true)
|-- MultipleLines: string (nullable = true)
|-- InternetService: string (nullable = true)
|-- OnlineSecurity: string (nullable = true)
|-- OnlineBackup: string (nullable = true)
|-- DeviceProtection: string (nullable = true)
|-- TechSupport: string (nullable = true)
|-- StreamingTV: string (nullable = true)
|-- StreamingMovies: string (nullable = true)
|-- Contract: string (nullable = true)
|-- PaperlessBilling: string (nullable = true)
|-- PaymentMethod: string (nullable = true)
|-- MonthlyCharges: double (nullable = true)
|-- TotalCharges: string (nullable = true)
|-- Churn: string (nullable = true)
```

Some of these features include numerical and categorical attributes such as gender, tenure, internet service, device protection, monthly and total charges. The target variable for our classification models is churn which is either a 'Yes' or 'No.' Through the model development phase, we eventually convert these attributes into binary values. Our main task is to analyze this dataset and predict the user churn rate. This type of analysis can allow the telecommunication company to further identify customers who will or will not churn (renew their contract) and proactively strategize in the methods of incentives, customer satisfaction and bundle deals for different services. The excel file is shown below for reference:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBill	PaymentMethod	MonthlyCharges	TotalCharges	Churn
2	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes	No	No	No	No	Month-tc	Yes	Electronic check	29.85	29.85	No
3	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes	No	No	No	One year	No	Mailed check	56.95	1889.5	No
4	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	No	No	No	No	Month-tc	Yes	Mailed check	53.85	108.15	Yes
5	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	No	Yes	Yes	No	No	One year	No	Bank transfer (au	42.3	1840.75	No
6	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	No	No	No	No	No	Month-tc	Yes	Electronic check	70.7	151.65	Yes
7	9305-CDSKC	Female	0	No	No	8	Yes	Yes	Fiber optic	No	No	Yes	No	Yes	Yes	Month-tc	Yes	Electronic check	99.65	820.5	Yes
8	1452-KIOVK	Male	0	No	Yes	22	Yes	Yes	Fiber optic	No	Yes	No	No	Yes	No	Month-tc	Yes	Credit card (auto	89.1	1949.4	No
9	6713-OKOM	Female	0	No	No	10	No	No phone service	DSL	Yes	No	No	No	No	No	Month-tc	No	Mailed check	29.75	301.9	No
10	7892-POOKP	Female	0	Yes	No	28	Yes	Yes	Fiber optic	No	No	Yes	Yes	Yes	Yes	Month-tc	Yes	Electronic check	104.8	3046.05	Yes
11	6388-TABGU	Male	0	No	Yes	62	Yes	No	DSL	Yes	Yes	No	No	No	No	One year	No	Bank transfer (au	56.15	3487.95	No
12	9763-GRSKD	Male	0	Yes	Yes	13	Yes	No	DSL	Yes	No	No	No	No	No	Month-tc	Yes	Mailed check	49.95	587.45	No
13	7469-LKBCI	Male	0	No	No	16	Yes	No	No	No internet ser	No internet se	No internet servi	No internet s	No internet se	No internet service	Two year	No	Credit card (auto	18.95	326.8	No
14	8091-TTVAX	Male	0	Yes	No	58	Yes	Yes	Fiber optic	No	No	Yes	No	Yes	Yes	One year	No	Credit card (auto	100.35	5681.1	No
15	0280-XJGEX	Male	0	No	No	49	Yes	Yes	Fiber optic	No	Yes	Yes	No	Yes	Yes	Month-tc	Yes	Bank transfer (au	103.7	5036.3	Yes
16	5129-JLPIS	Male	0	No	No	25	Yes	No	Fiber optic	Yes	No	Yes	Yes	Yes	Yes	Month-tc	Yes	Electronic check	105.5	2686.05	No
17	3655-SNQYZ	Female	0	Yes	Yes	69	Yes	Yes	Fiber optic	Yes	Yes	Yes	Yes	Yes	Yes	Two year	No	Credit card (auto	113.25	7895.15	No
18	8191-XWSZG	Female	0	No	No	52	Yes	No	No	No internet ser	No internet se	No internet servi	No internet s	No internet se	No internet service	One year	No	Mailed check	20.65	1022.95	No
19	9959-WOFKT	Male	0	No	Yes	71	Yes	Yes	Fiber optic	Yes	No	Yes	No	Yes	Yes	Two year	No	Bank transfer (au	106.7	7382.25	No
20	4190-MFLUV	Female	0	Yes	Yes	10	Yes	No	DSL	No	No	Yes	Yes	No	No	Month-tc	No	Credit card (auto	55.2	528.35	Yes
21	4183-MYFRB	Female	0	No	No	21	Yes	No	Fiber optic	No	Yes	Yes	No	No	Yes	Month-tc	Yes	Electronic check	90.05	1862.9	No
22	8779-QRDM	Male	1	No	No	1	No	No phone service	DSL	No	No	Yes	No	No	Yes	Month-tc	Yes	Electronic check	39.65	39.65	Yes
23	1680-VDCW	Male	0	Yes	No	12	Yes	No	No	No internet ser	No internet se	No internet servi	No internet s	No internet se	No internet service	One year	No	Bank transfer (au	19.8	202.25	No
24	1066-JKSGK	Male	0	No	No	1	Yes	No	No	No internet ser	No internet se	No internet servi	No internet s	No internet se	No internet service	Month-tc	No	Mailed check	20.15	20.15	Yes
25	3638-WEAB	Female	0	Yes	No	58	Yes	Yes	DSL	No	Yes	No	Yes	No	No	Two year	Yes	Credit card (auto	59.9	3505.1	No
26	6322-HRPFA	Male	0	Yes	Yes	49	Yes	No	DSL	Yes	Yes	No	Yes	No	No	Month-tc	No	Credit card (auto	59.6	2970.3	No
27	6865-JZNKO	Female	0	No	No	30	Yes	No	DSL	Yes	Yes	No	No	No	No	Month-tc	Yes	Bank transfer (au	55.3	1530.6	No
28	6467-CHFZW	Male	0	Yes	Yes	47	Yes	Yes	Fiber optic	No	Yes	No	No	Yes	Yes	Month-tc	Yes	Electronic check	99.35	4749.15	Yes
29	8665-UTDHZ	Male	0	Yes	Yes	1	No	No phone service	DSL	No	Yes	No	No	No	No	Month-tc	No	Electronic check	30.2	30.2	Yes
30	5248-YGIJN	Male	0	Yes	No	72	Yes	Yes	DSL	Yes	Yes	Yes	Yes	Yes	Yes	Two year	Yes	Credit card (auto	90.25	6369.45	No
31	8773-HHUOZ	Female	0	No	Yes	17	Yes	No	DSL	No	No	No	No	Yes	Yes	Month-tc	Yes	Mailed check	64.7	1093.1	Yes
32	3841-NFECX	Female	1	Yes	No	71	Yes	Yes	Fiber optic	Yes	Yes	Yes	Yes	No	No	Two year	Yes	Credit card (auto	96.35	6766.95	No
33	4929-XIHVW	Male	1	Yes	No	2	Yes	No	Fiber optic	No	No	Yes	No	Yes	Yes	Month-tc	Yes	Credit card (auto	95.5	181.65	No
34	6827-IEAUQ	Female	0	Yes	Yes	27	Yes	No	DSL	Yes	Yes	Yes	Yes	No	No	One year	No	Mailed check	66.15	1874.45	No
35	7310-EGVHZ	Male	0	No	No	1	Yes	No	No	No internet ser	No internet se	No internet servi	No internet s	No internet se	No internet service	Month-tc	No	Bank transfer (au	20.2	20.2	No

Moreover, the visualization below shows some basic dataset statistics and insights when we were in our Exploratory Data Analysis (EDA) phase.

Dataset Statistics

Number of Variables	21
Number of Rows	5986
Missing Cells	0
Missing Cells (%)	0.0%
Duplicate Rows	0
Duplicate Rows (%)	0.0%
Total Size in Memory	6.7 MB
Average Row Size in Memory	1.1 KB
Variable Types	Categorical: 18 Numerical: 3

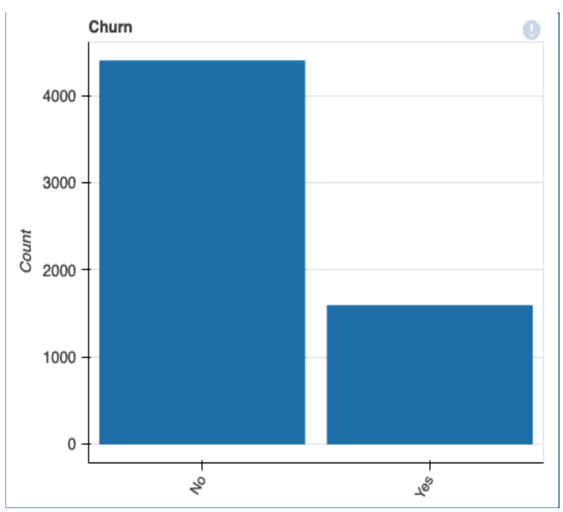
Dataset Insights

`SeniorCitizen` is skewed	Skewed
`tenure` is skewed	Skewed
`MonthlyCharges` is skewed	Skewed
`customerID` has a high cardinality: 5986 distinct values	High Cardinality
`TotalCharges` has a high cardinality: 5611 distinct values	High Cardinality
`customerID` has constant length 10	Constant Length
`customerID` has all distinct values	Unique
`SeniorCitizen` has 5020 (83.86%) zeros	Zeros

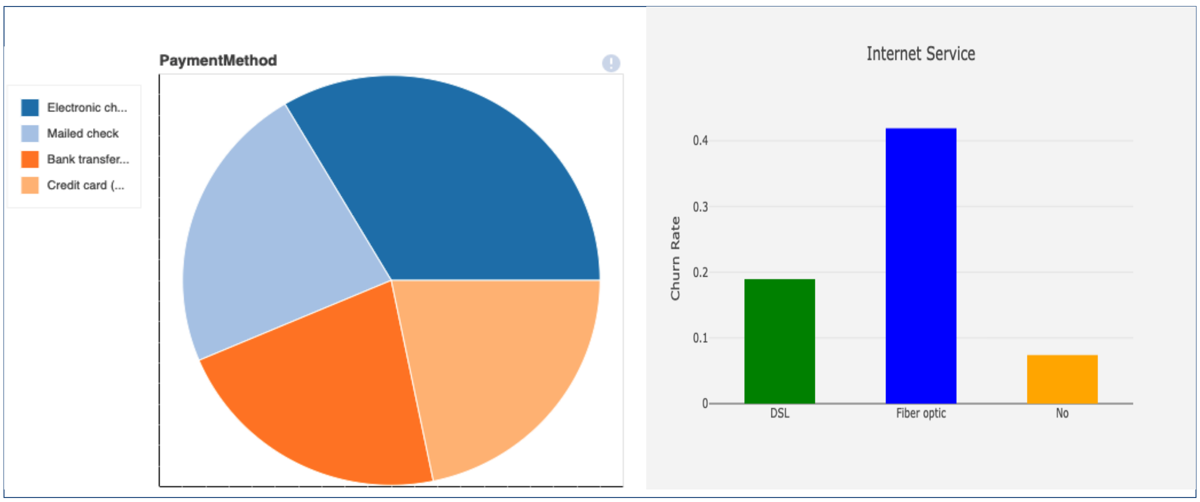
There are a total of 21 features and 5986 rows, indicating that for each customer there are 21 attributes as data points. Since we got our dataset from Kaggle, we see that there are 0 missing or null values. The dataset is intended for educational purposes and therefore is evidently very clean. The dataset insights portion of the visualization above displays some distribution aspects of various features. For example, the senior citizen feature has 83.86% zeros which indicates that the customer base is composed of majority non senior citizens. Moving forward with the EDA process, which allowed us to gain a better statistical understanding of the features and their importance before feeding them into the 3 models to predict churn, we discovered that approximately 25% of the dataset has churned or that they did not renew their contract. The visualization below also shows that females and males are approximately equally likely to churn with females having a slightly higher churn rate.

```
df_data.groupby('gender').Churn.mean()
```

```
gender
Female    0.269209
Male      0.261603
Name: Churn, dtype: float64
```



Lastly, customers with fiber optic internet service have the highest churn rate among internet service categories which could be due to prices, competition from other providers and other factors. Also, we broke down the payment method feature which results in discovering that most customers pay by electronic check. The other 3 methods, check mailed, bank transfer and credit card, all seem to occur approximately the same frequency. The 2 visualizations below display these two insights:



III. Data Preprocessing and Data Modeling

Moving further from data understanding to our main goal: exploring impact of scale on quality of analysis and performance, parallel computation, the data need to be preprocessed. In the preprocessing, Pyspark was implemented to build the pipeline. The numerical features of the dataset were loaded into vectorAssembler and categorical features were passed through stringIndexer and one-hot encoding. After those conversions, a new column 'features' which stores a vectorized list of binary values were added to the dataset. Finally, the dataset was fitted to the train split and transform both the training and test splits (70/30 random split). Logistic regression, Decision tree, and SVC models were used to perform binary classification.

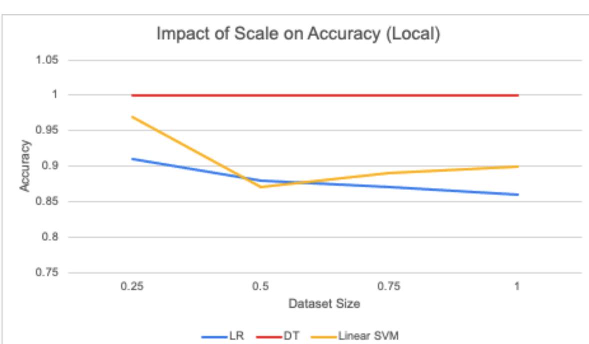
```
   label                                           features  ... TotalCharges  Churn
0    0.0  (1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, ...  ...      1734.65     No
1    0.0  (0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 1.0, ...  ...       3973.2     No
2    1.0  (0.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, ...  ...      2869.85    Yes
3    0.0  (1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, ...  ...        238.5     No
4    0.0  (1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, ...  ...        119.5     No

[5 rows x 22 columns]
```

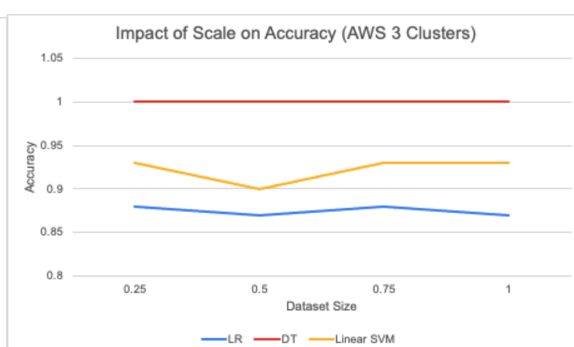
IV. Performance Measures

In order to explore impacts of scale on quality of analysis, impact of scale on time performance, and impact of parallel computation on performance with respect to scalability, Logistic regression, Decision Tree, Linear Support Vector Machine models were implemented in both local and distributed parallel environment. To measure performances, the accuracy of each model and time consumption has been measured with respect to randomized proportions of dataset.

a. Scale vs Classification Accuracy Metrics

Since the telecom churn data is relatively small with roughly 7,000 rows and considered as ‘clean’ dataset (which has no noises), high accuracies on each model were evident. To measure impact of scale on quality of analysis, 3 classification models were deployed on different scales of data: 25%, 50%, 75%, and 100%. The following graph represents the model performances on different data scales in local environment and AWS clusters.

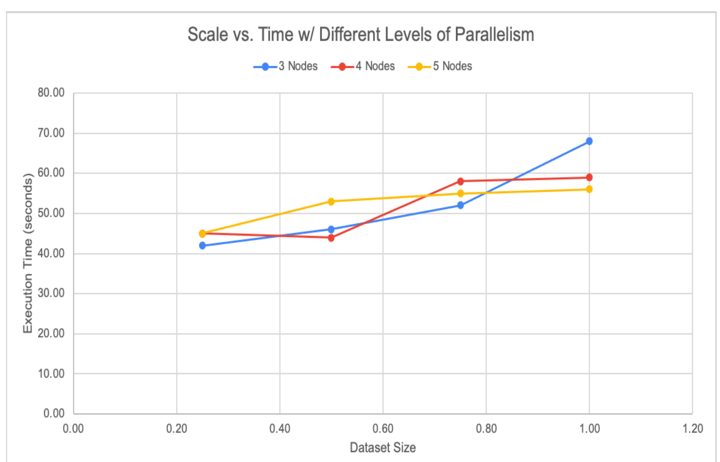




In the local environment, the highest accuracy is achieved with least dataset size. This contradicts the normal belief that more data leads to higher accuracy scores. This phenomenon may have been caused by the dataset's clean trait, which may imply that the smaller clean dataset could result better accuracy. However, the accuracy metrics in 2 models from local and AWS environments does not show a sizable difference. Scaling in this small dataset does not affect the quality of the model significantly. In order to delve further to see a sizeable difference in accuracy metrics, a larger dataset should be needed to discover deeper findings on impact of scale on accuracy. The major eye-catching anomaly is that our dataset had 100% accuracy across various scales on both environments. This was unexpected as there were no false positives or false negatives. This accuracy performance could either be the result of a simple dataset and small number of numerical features.

b. Scale vs. Time

To test the impact of scale on time performance, 3 models were executed on local machine and on the AWS cluster with various size cluster. While it is evident that there is a positive relationship between time consumption and dataset size, the figure implies that there is a negative relation between number of clusters and time performance. Even though the time takes longer with 4 clusters takes longer than 3 clusters in 75% scale, the status of showing better performance with more clusters are evident in larger scale such as 100% scale. Furthermore, the trend of more data taking more time was clearly

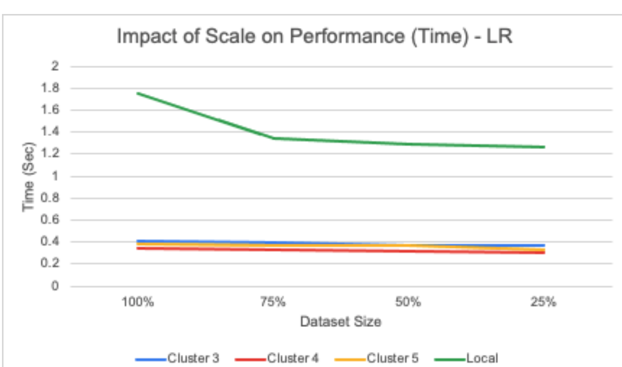


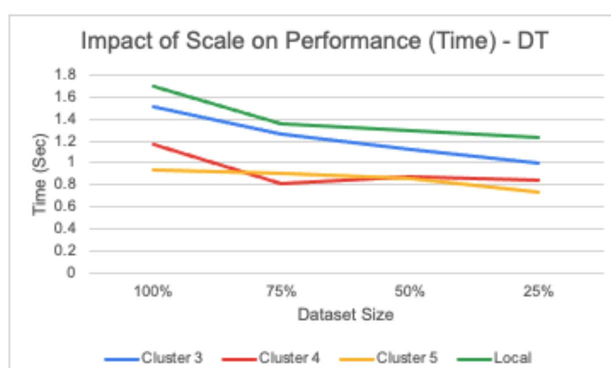
visible taking into consideration the variable traffic that was faced on the cluster network during this testing.

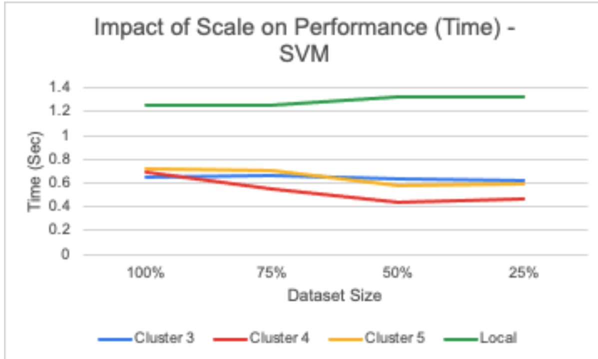
The parallel computing environment showed better performance than local computing as the run time was pretty noticeable jumping from local to cluster computing. Despite the fact that our dataset was on the smaller side, the difference between the run time of different size of datasets was very noticeable. In the case of running logistic regression model to the dataset, local machines tend to take more than a second to run while AWS cluster environment took less than 0.5 seconds in every dataset size. Furthermore, time vs. scaling plot tends to be flatted as current dataset may not be big enough to reveal more significant effect of scaling on runtime among different clusters.

Execution Time (LR)	Cluster Size			
Dataset Size	Cluster 3	Cluster 4	Cluster 5	Local
100%	0.411	0.349	0.383	1.75
75%	0.391	0.324	0.365	1.34
50%	0.375	0.314	0.365	1.29
25%	0.372	0.305	0.333	1.26

Table 1. Run Time Table for Logistic Regression on AWS cluster







c. Model Comparison

By referencing various performance measures, it is evident that local non-parallel environment consumes more time than parallel computing environment in every classification models. Hence, parallel computing environment is much more ideal than local machine as the dataset increases. In such case where there are merely less than 10,000 rows of dataset, the difference on time performance would not be noticeable by users.

Furthermore, as the cluster size increases in parallel environment, run-time is generally decreased. The negative relationship between number of clusters and run-time incentivizes users to use more clusters than less to derive outcomes faster.

The Decision Tree model resulted in 100% accuracy in various dataset size on both local and parallel environment. The clean and simple data traits which resulted not enough data to add complexity and remove bias from the model. Oddly, there was no sign of over-fitting by examining the results.

V. Conclusion

Customer churn prediction is widely used by telecommunication company to analyze the traits of opting out and empower services to have comparative advantages over competitors. This study has reflected the importance of churn prediction and performed a predictive analysis of telecommunications customer data on local and AWS parallel environment. The performance metrics clearly shows that there is significant advantage of using parallel computing environment respect to time performance. Even though our dataset was small- and the-time difference was miniscule, the study incentivizes telecom stakeholders to implement churn/classification prediction to a large real-world dataset with parallel environment. Additionally, the future work of this study would involve collecting a larger dataset to empower examination on impact of scale on quality of analysis.

Appendix

```
In [2]: import pyspark
        from pyspark.sql import SQLContext
        from pyspark.conf import SparkConf
        from pyspark.sql import SparkSession
        from pyspark.sql.types import DoubleType
        from pyspark.sql.types import DecimalType
        from pyspark.sql.types import IntegerType
        from pyspark.sql.types import LongType
        from pyspark.sql.functions import UserDefinedFunction
        from pyspark.ml.feature import OneHotEncoderEstimator, StringIndexer, VectorIndexer, IndexToString, VectorAssembler
        from pyspark.ml import Pipeline
        from pyspark.ml.feature import VectorAssembler
        from pyspark.ml.classification import RandomForestClassifier
        from pyspark.ml.evaluation import BinaryClassificationEvaluator
```

```
sc.install_pypi_package("pandas==0.25.1")
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…

Collecting pandas==0.25.1
  Using cached pandas-0.25.1-cp36-cp36m-manylinux1_x86_64.whl (10.5 MB)
Collecting python-dateutil>=2.6.1
  Using cached python_dateutil-2.8.1-py2.py3-none-any.whl (227 kB)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib64/python3.6/site-packages (from pandas==0.25.1) (1.14.
5)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/site-packages (from pandas==0.25.1) (2019.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/site-packages (from python-dateutil>=2.6.1->panda
s==0.25.1) (1.12.0)
Installing collected packages: python-dateutil, pandas
Successfully installed pandas-0.25.1 python-dateutil-2.8.1

Collecting numpy==1.15.4
  Downloading numpy-1.15.4-cp36-cp36m-manylinux1_x86_64.whl (13.9 MB)
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 1.14.5
    Not uninstalling numpy at /usr/local/lib64/python3.6/site-packages, outside environment /tmp/1620083632056-0
    Can't uninstall 'numpy'. No files were found to uninstall.
Successfully installed numpy-1.15.4


ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behav
iour is the source of the following dependency conflicts.
python36-sagemaker-pyspark 1.2.4 requires pyspark==2.3.2, which is not installed.
```

```
import pandas as pd
import numpy as np
from pyspark import SparkContext
import pyspark.sql
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

```
data = sqlContext.read.load('s3://aws-logs-862133200906-us-east-1/elasticmapreduce/telecom_users.csv',format='com.datab
```

```
---------------------------------------------------------------------------
```

```
: data.printSchema()
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…

root
 |-- _c0: integer (nullable = true)
 |-- customerID: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- SeniorCitizen: integer (nullable = true)
 |-- Partner: string (nullable = true)
 |-- Dependents: string (nullable = true)
 |-- tenure: integer (nullable = true)
 |-- PhoneService: string (nullable = true)
 |-- MultipleLines: string (nullable = true)
 |-- InternetService: string (nullable = true)
 |-- OnlineSecurity: string (nullable = true)
 |-- OnlineBackup: string (nullable = true)
 |-- DeviceProtection: string (nullable = true)
 |-- TechSupport: string (nullable = true)
 |-- StreamingTV: string (nullable = true)
 |-- StreamingMovies: string (nullable = true)
 |-- Contract: string (nullable = true)
 |-- PaperlessBilling: string (nullable = true)
 |-- PaymentMethod: string (nullable = true)
 |-- MonthlyCharges: double (nullable = true)
 |-- TotalCharges: string (nullable = true)
 |-- Churn: string (nullable = true)
```

```
: data = data.drop('_c0').drop('customerID')
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

```
: cols = data.columns
cols
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

```
In [50]: pd.DataFrame(data.take(5), columns = data.columns)
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…

   gender  SeniorCitizen Partner  ... MonthlyCharges  TotalCharges Churn
0    Male              0     Yes  ...          24.10       1734.65    No
1  Female              0      No  ...          88.15        3973.2    No
2  Female              1     Yes  ...          74.95       2869.85   Yes
3    Male              0      No  ...          55.90         238.5    No
4    Male              0      No  ...          53.45         119.5    No

[5 rows x 20 columns]
```

```
In [51]: data.groupby('StreamingMovies').count().toPandas()
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…

       StreamingMovies  count
0                   No   2356
1                  Yes   2339
2  No internet service   1291
```

```
In [52]: Categorical_ = [item[0] for item in data.dtypes if item[1].startswith('string')]
         Numerical_ = [item[0] for item in data.dtypes if item[1] == 'int' or item[1] == 'double']
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

```
In [53]: stages=[]
         for categoricalCol in Categorical_:
             stringIndexer = StringIndexer(inputCol = categoricalCol, outputCol = categoricalCol + 'Index')
             encoder = OneHotEncoderEstimator(inputCols=[stringIndexer.getOutputCol()], outputCols = [categoricalCol + 'classVec
             stages += [stringIndexer, encoder]
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

In [54]:
```
label_stringIdx = StringIndexer(inputCol = 'Churn', outputCol = 'label')
stages += [label_stringIdx]
assemblerInputs = [c + 'classVec' for c in Categorical_] + Numerical_
assembler = VectorAssembler(inputCols = assemblerInputs, outputCol = 'features')
stages += [assembler]
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

In [55]:
```
pipeline = Pipeline(stages = stages)
pipelineModel = pipeline.fit(data)
data = pipelineModel.transform(data)
selectedCols = ['label', 'features'] + cols
data = data.select(selectedCols)
data.printSchema()
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…

root
 |-- label: double (nullable = false)
 |-- features: vector (nullable = true)
 |-- gender: string (nullable = true)
 |-- SeniorCitizen: integer (nullable = true)
 |-- Partner: string (nullable = true)
 |-- Dependents: string (nullable = true)
 |-- tenure: integer (nullable = true)
 |-- PhoneService: string (nullable = true)
 |-- MultipleLines: string (nullable = true)
 |-- InternetService: string (nullable = true)
 |-- OnlineSecurity: string (nullable = true)
 |-- OnlineBackup: string (nullable = true)
 |-- DeviceProtection: string (nullable = true)
 |-- TechSupport: string (nullable = true)
 |-- StreamingTV: string (nullable = true)
 |-- StreamingMovies: string (nullable = true)
 |-- Contract: string (nullable = true)
 |-- PaperlessBilling: string (nullable = true)
 |-- PaymentMethod: string (nullable = true)
 |-- MonthlyCharges: double (nullable = true)
 |-- TotalCharges: string (nullable = true)
 |-- Churn: string (nullable = true)
```

In [56]:
```
data.toPandas().head(5)
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…

   label                                           features  ... TotalCharges  Churn
0    0.0  (1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, ...  ...      1734.65     No
1    0.0  (0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 1.0, ...  ...       3973.2     No
2    1.0  (0.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, ...  ...      2869.85    Yes
3    0.0  (1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, ...  ...        238.5     No
4    0.0  (1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, ...  ...        119.5     No

[5 rows x 22 columns]
```

In [114]:
```
# data scale testing
scaled_data = data
#scaled_data, x = data.randomSplit([0.75,0.25], seed = 123)
#scaled_data, x = data.randomSplit([0.50,0.50], seed = 123)
#scaled_data, x = data.randomSplit([0.25,0.75], seed = 123)

train, test = scaled_data.randomSplit([0.7,0.3], seed = 123)
print("Training Dataset :" + str(train.count()))
print("Testing Dataset :" + str(test.count()))
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…

Training Dataset :4179
Testing Dataset :1807
```

In [115]:
```
from pyspark.ml.classification import LogisticRegression
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

In [116]:
```
lr = LogisticRegression(featuresCol = 'features', labelCol = 'label', maxIter = 2)
lrModel = lr.fit(train)
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

In [117]:
```
lr_summary = lrModel.summary
print("area under ROC :"+str(lr_summary.areaUnderROC))
print("precision :"+str(lr_summary.precisionByLabel))
print("recall :"+str(lr_summary.recallByLabel))
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…

area under ROC :0.9996264008864048
precision :[0.9499072356215214, 1.0]
recall :[1.0, 0.8536585365853658]
```

In [118]:
```
pred = lrModel.transform(test)
label = pred.select('label').toPandas()
prediction = pred.select('prediction').toPandas()
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

In [119]:
```
#sc.install_pypi_package("sklearn")
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

In [120]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

print(str(confusion_matrix(label, prediction)))
print(str(accuracy_score(label,prediction)))
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…

[[1323    4]
 [ 230  250]]
0.8705035971223022
```

In [121]:
```
from pyspark.ml.classification import DecisionTreeClassifier
```
```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

In [122]:
```
dt = DecisionTreeClassifier(labelCol="label", featuresCol="features")
```
```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

In [123]:
```
dt_model = dt.fit(train)
```
```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

In [124]:
```
predictions = dt_model.transform(test)
```
```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

In [125]:
```
label = predictions.select('label').toPandas()
prediction = predictions.select('prediction').toPandas()
```
```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

In [126]:
```
print(str(confusion_matrix(label, prediction)))
print(str(accuracy_score(label,prediction)))
```
```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…

[[1327    0]
 [   0  480]]
1.0
```

In [127]:
```
from pyspark.ml.classification import LinearSVC
lsvc = LinearSVC(maxIter=10, regParam=0.1)
```
```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

In [133]:
```
lsvcModel = lsvc.fit(train.select(['label','features']))
```
```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

In [134]:
```
predictions = lsvcModel.transform(test)
```
```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

In [135]:
```
label = predictions.select('label').toPandas()
prediction = predictions.select('prediction').toPandas()
```
```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

In [136]:
```
print(str(confusion_matrix(label, prediction)))
print(str(accuracy_score(label,prediction)))
```
```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…

[[1320    7]
 [ 104  376]]
0.9385722191477587
```

TELECOM CHURN PREDICTION

Importing Libraries

In [9]:
```
import pyspark
import pandas as pd
import numpy as np
from pyspark.sql import SQLContext
from pyspark.conf import SparkConf
from pyspark.sql import SparkSession
from pyspark.sql.types import DoubleType
from pyspark.sql.types import DecimalType
from pyspark.sql.types import IntegerType
from pyspark.sql.types import LongType
from pyspark.sql.functions import UserDefinedFunction
from pyspark.ml.feature import OneHotEncoder, StringIndexer, VectorIndexer, IndexToString, VectorAssembler
from pyspark.ml import Pipeline
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import BinaryClassificationEvaluator
```

In [2]:
```
pip install pyspark
```

```
Collecting pyspark
  Downloading pyspark-3.1.1.tar.gz (212.3 MB)
     |████████████████████████████████| 212.3 MB 30.3 MB/s eta 0:00:01
Collecting py4j==0.10.9
  Downloading py4j-0.10.9-py2.py3-none-any.whl (198 kB)
     |████████████████████████████████| 198 kB 31.0 MB/s eta 0:00:01
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.1.1-py2.py3-none-any.whl size=212767604 sha256=c648d9775a945d6f23dcb1
ae06e1c866b566f72beaec6ad86a849e64b6e0e00f
  Stored in directory: /Users/junghopark/Library/Caches/pip/wheels/43/47/42/bc413c760cf9d3f7b46ab7cd6590e8c47ebfd19a7
386cd4a57
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9 pyspark-3.1.1
Note: you may need to restart the kernel to use updated packages.
```

In [16]:
```
import os
os.environ['PYSPARK_SUBMIT_ARGS'] = "--master mymaster --total-executor 2 --conf "spark.driver.extraJavaOptions=-Dhttp.
```

In [14]:
```
from pyspark import SparkContext
import pyspark.sql

sc = SparkContext(appName="PythonStreamingQueueStream")
sqlContext = SQLContext(sc)
```

In [3]:
```
data = sqlContext.read.load('telecom_users.csv',format='com.databricks.spark.csv', header = True, inferSchema='true')
```

In [4]:
```
data.printSchema()
```

```
root
 |-- _c0: integer (nullable = true)
 |-- customerID: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- SeniorCitizen: integer (nullable = true)
 |-- Partner: string (nullable = true)
 |-- Dependents: string (nullable = true)
 |-- tenure: integer (nullable = true)
 |-- PhoneService: string (nullable = true)
 |-- MultipleLines: string (nullable = true)
 |-- InternetService: string (nullable = true)
 |-- OnlineSecurity: string (nullable = true)
 |-- OnlineBackup: string (nullable = true)
 |-- DeviceProtection: string (nullable = true)
 |-- TechSupport: string (nullable = true)
 |-- StreamingTV: string (nullable = true)
 |-- StreamingMovies: string (nullable = true)
 |-- Contract: string (nullable = true)
 |-- PaperlessBilling: string (nullable = true)
 |-- PaymentMethod: string (nullable = true)
 |-- MonthlyCharges: double (nullable = true)
 |-- TotalCharges: string (nullable = true)
 |-- Churn: string (nullable = true)
```

In [5]:
```
data = data.drop('_c0').drop('customerID')
```

In [6]:
```
cols = data.columns
cols
```

Out[6]:
```
['gender',
 'SeniorCitizen',
 'Partner',
 'Dependents',
 'tenure',
 'PhoneService',
 'MultipleLines',
 'InternetService',
 'OnlineSecurity',
 'OnlineBackup',
 'DeviceProtection',
 'TechSupport',
 'StreamingTV',
 'StreamingMovies',
 'Contract',
 'PaperlessBilling',
 'PaymentMethod',
 'MonthlyCharges',
 'TotalCharges',
 'Churn']
```

In [7]:
```
pd.DataFrame(data.take(5), columns = data.columns)
```

Out[7]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupp
0	Male	0	Yes	Yes	72	Yes	Yes	No	No internet service	No internet service	No internet service	No inter serv
1	Female	0	No	No	44	Yes	No	Fiber optic	No	Yes	Yes	
2	Female	1	Yes	No	38	Yes	Yes	Fiber optic	No	No	No	
3	Male	0	No	No	4	Yes	No	DSL	No	No	No	
4	Male	0	No	No	2	Yes	No	DSL	Yes	No	Yes	

```
In [8]: data.groupby('StreamingMovies').count().toPandas()
```

Out[8]:

	StreamingMovies	count
0	No	2356
1	Yes	2339
2	No internet service	1291

```
In [9]: Categorical_ = [item[0] for item in data.dtypes if item[1].startswith('string')]
Numerical_ = [item[0] for item in data.dtypes if item[1] == 'int' or item[1] == 'double']
```

```
In [10]: stages=[]
for categoricalCol in Categorical_:
    stringIndexer = StringIndexer(inputCol = categoricalCol, outputCol = categoricalCol + 'Index')
    encoder = OneHotEncoder(inputCols=[stringIndexer.getOutputCol()], outputCols = [categoricalCol + 'classVec'])
    stages += [stringIndexer, encoder]
label_stringIdx = StringIndexer(inputCol = 'Churn', outputCol = 'label')
stages += [label_stringIdx]
assemblerInputs = [c + 'classVec' for c in Categorical_] + Numerical_
assembler = VectorAssembler(inputCols = assemblerInputs, outputCol = 'features')
stages += [assembler]
```

```
In [11]: pipeline = Pipeline(stages = stages)
pipelineModel = pipeline.fit(data)
data = pipelineModel.transform(data)
selectedCols = ['label', 'features'] + cols
data = data.select(selectedCols)
data.printSchema()
```

```
root
 |-- label: double (nullable = false)
 |-- features: vector (nullable = true)
 |-- gender: string (nullable = true)
 |-- SeniorCitizen: integer (nullable = true)
 |-- Partner: string (nullable = true)
 |-- Dependents: string (nullable = true)
 |-- tenure: integer (nullable = true)
 |-- PhoneService: string (nullable = true)
 |-- MultipleLines: string (nullable = true)
 |-- InternetService: string (nullable = true)
 |-- OnlineSecurity: string (nullable = true)
```

```
In [12]: data.toPandas().head(5)
```

Out[12]:

	label	features	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	...	DeviceProtection	TechSupport	Stream
0	0.0	(1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, ...	Male	0	Yes	Yes	72	Yes	Yes	No	...	No internet service	No internet service	No ir s
1	0.0	(0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 1.0, ...	Female	0	No	No	44	Yes	No	Fiber optic	...	Yes	No	
2	1.0	(0.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, ...	Female	1	Yes	No	38	Yes	Yes	Fiber optic	...	No	No	
3	0.0	(1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, ...	Male	0	No	No	4	Yes	No	DSL	...	No	No	
4	0.0	(1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, ...	Male	0	No	No	2	Yes	No	DSL	...	Yes	No	

5 rows × 22 columns

```
In [13]: train, test = data.randomSplit([0.1,0.9], seed = 472)
print("Training Dataset :" + str(train.count()))
print("Testing Dataset :" + str(test.count()))
```

```
Training Dataset :610
Testing Dataset :5376
```

In [14]:
```
%%time
from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression(featuresCol = 'features', labelCol = 'label', maxIter = 2)
lrModel = lr.fit(train)
```

```
Wall time: 1.84 s
```

In [15]:
```
lr_summary = lrModel.summary
```

In [16]:
```
lr_summary.accuracy
```

Out[16]:
```
0.940983606557377
```

In [17]:
```
lr_summary.areaUnderROC
```

Out[17]:
```
0.9999856354860952
```

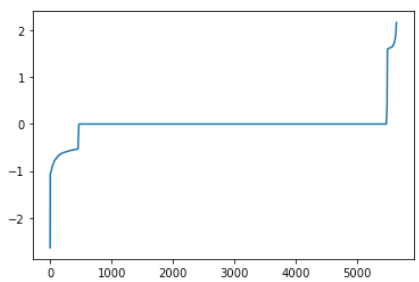
In [18]:
```
lr_summary.precisionByLabel
```

Out[18]:
```
[0.9271255060728745, 1.0]
```

In [19]:
```
lr_summary.recallByLabel
```

Out[19]:
```
[1.0, 0.7631578947368421]
```

In [20]:
```
import matplotlib.pyplot as plt
beta = np.sort(lrModel.coefficients)
plt.plot(beta)
plt.show()
```



In [21]:
```
pred = lrModel.transform(test)
pred.select('label','prediction').toPandas()
```

Out[21]:

	label	prediction
0	0.0	0.0
1	0.0	0.0
2	0.0	0.0
3	0.0	0.0
4	0.0	0.0
...	...	...
5371	1.0	0.0
5372	1.0	0.0
5373	1.0	0.0
5374	1.0	0.0
5375	1.0	0.0

5376 rows × 2 columns

```
In [22]: label = pred.select('label').toPandas()
         prediction = pred.select('prediction').toPandas()

In [23]: from sklearn.metrics import confusion_matrix
         confusion_matrix(label, prediction)

Out[23]: array([[3941,    0],
                [1113,  322]], dtype=int64)

In [24]: from sklearn.metrics import accuracy_score

         accuracy_score(label,prediction)

Out[24]: 0.79296875

In [25]: from pyspark.ml.classification import DecisionTreeClassifier

In [26]: dt = DecisionTreeClassifier(labelCol="label", featuresCol="features")

In [27]: dt_model = dt.fit(train)

In [28]: predictions = dt_model.transform(test)

In [29]: label = predictions.select('label').toPandas()
         prediction = predictions.select('prediction').toPandas()

In [30]: from sklearn.metrics import accuracy_score

         accuracy_score(label,prediction)

Out[30]: 1.0

In [31]: from sklearn.metrics import confusion_matrix
         confusion_matrix(label, prediction)

Out[31]: array([[3941,    0],
                [   0, 1435]], dtype=int64)
```

```
In [32]: from pyspark.ml.classification import LinearSVC
         lsvc = LinearSVC(maxIter=10, regParam=0.1)

In [33]: lsvcModel = lsvc.fit(train)

In [34]: predictions = lsvcModel.transform(test)

In [35]: label = predictions.select('label').toPandas()
         prediction = predictions.select('prediction').toPandas()

In [36]: from sklearn.metrics import accuracy_score

         accuracy_score(label,prediction)

Out[36]: 0.9034598214285714

In [37]: svm.precisionByLabel
```