

CUSTOMER CHURN PREDICTION

Using ML in Dataiku

WHAT IS THE NEED OF A CUSTOMER CHURN PREDICTION MODEL ?

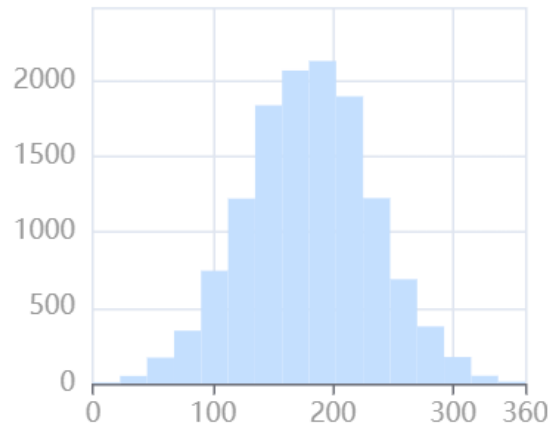
- Identify At-Risk Customers
- Reduce Revenue Loss
- Improve Customer Retention
- Enhance Customer Experience
- Gain Competitive Advantage

OBJECTIVES FOR THE CUSTOMER CHURN PREDICTION PROJECT

- Analyzing customer data and identifying behavioral patterns through Exploratory Data Analysis (EDA).
- To clean, preprocess, and engineer features from the raw dataset for optimal machine learning model performance.
- To apply K-Means Clustering for customer segmentation, enabling targeted retention strategies.
- To use a Random Forest model to predict which customers are likely to leave, based on patterns in the data.
- To visualize churn insights and customer trends within the Dataiku platform using charts, dashboards, and flow views.

▼ # 80_day_minutes 🗑️ ⋮

▼ Histogram

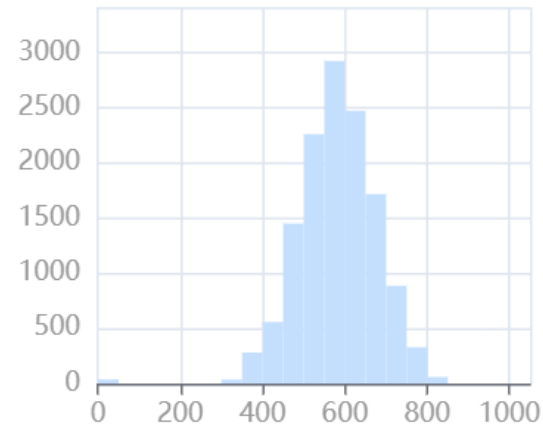


▼ Summary stats

N values	13009
N distinct	1442
N finite	13009
Mean	180.35645322
Median	180.5
Std Dev	52.955271659

▼ # 80_total_minutes 🗑️ ⋮

▼ Histogram

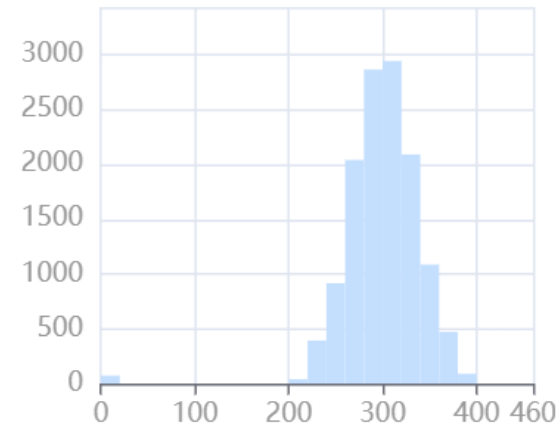


▼ Summary stats

N values	13009
N distinct	1747
N finite	13009
Mean	581.00103774
Median	583.2
Std Dev	93.992305867

▼ # 80_total_calls 🗑️ ⋮

▼ Histogram

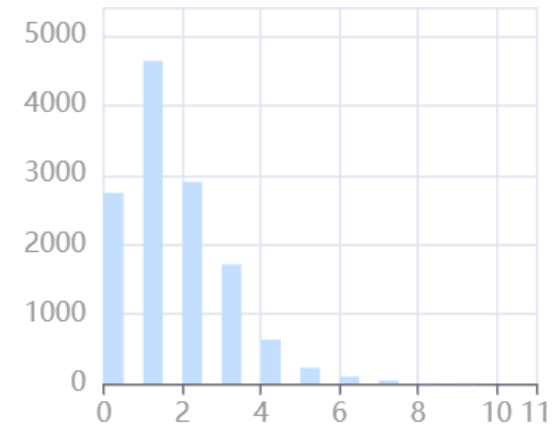


▼ Summary stats

N values	13009
N distinct	175
N finite	13009
Mean	299.37220386
Median	301
Std Dev	39.715642269

▼ # 80_Customer_ser... 🗑️ ⋮

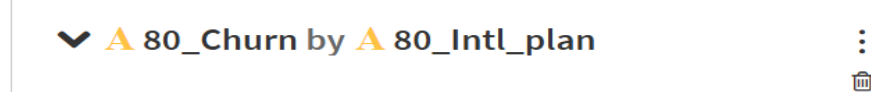
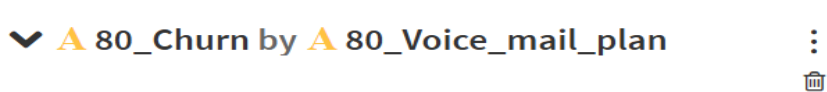
▼ Histogram



▼ Summary stats

N values	13009
N distinct	10
N finite	13009
Mean	1.5572296103
Median	1
Std Dev	1.3206551263

UNIVARIATE ANALYSIS ON NUMERIC FIELDS USING STATISTICS TAB

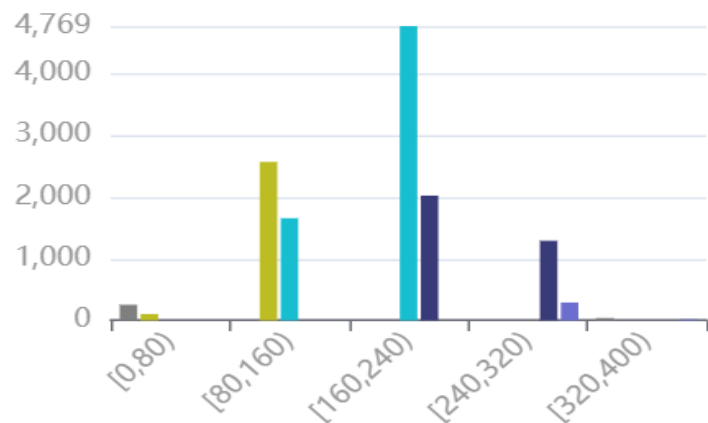


BIVARIATE ANALYSIS BETWEEN CATEGORICAL AND NUMERICAL FIELDS USING STATISTICS TAB

▼ # 80_day_charge by # 80_day_minutes split by 80_Churn

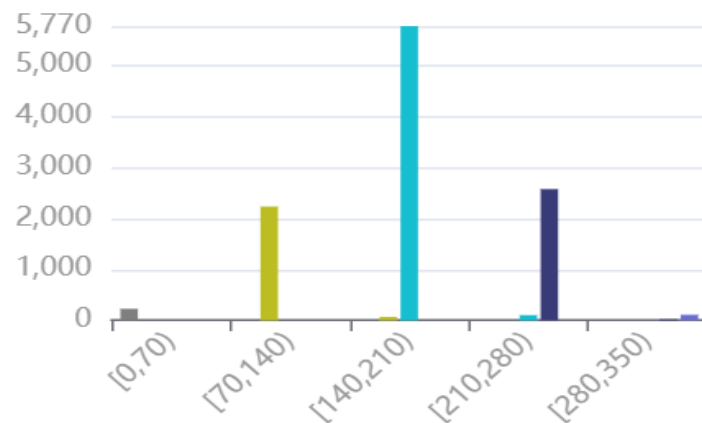
▼ All

▼ Histogram



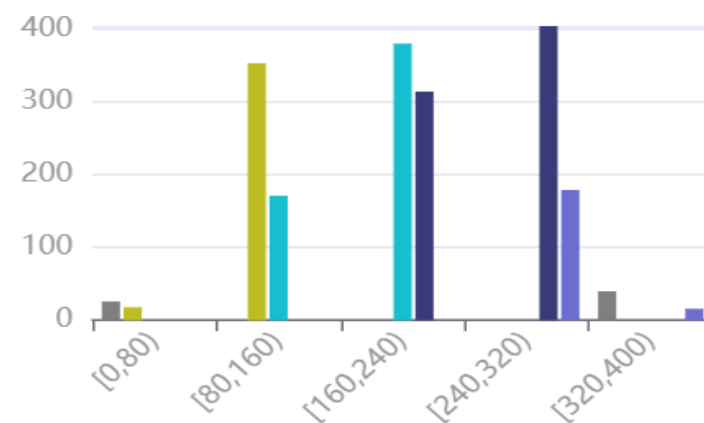
▼ False

▼ Histogram



▼ True

▼ Histogram



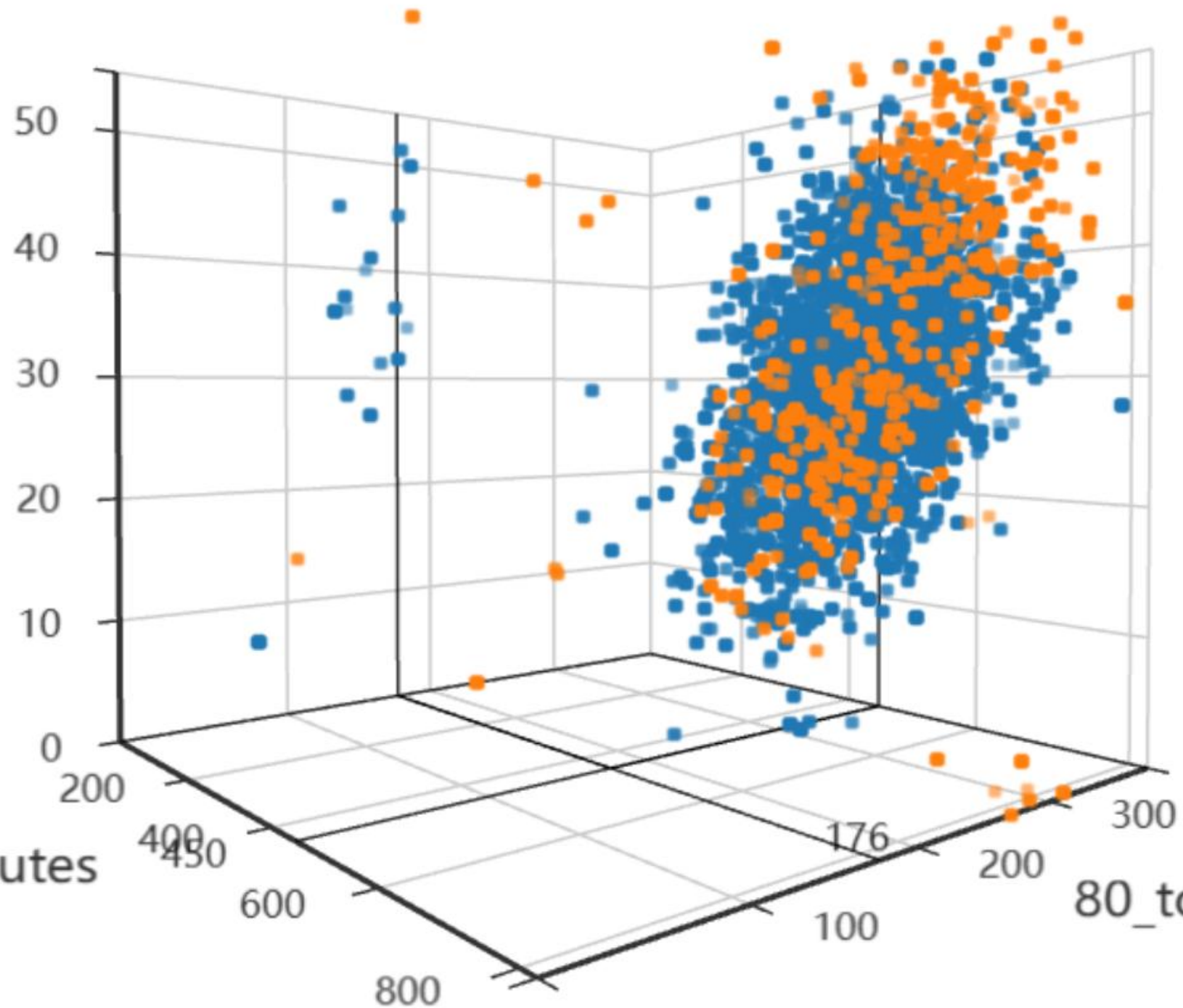
BIVARIATE ANALYSIS BETWEEN NUMERICAL FIELDS USING TARGET VARIABLE AS A SPLITTING FACTOR

**A Multivariate Analysis
on 3 variables
showcasing the
dependency of each
other and their impact
on the target variable
through a 3D Scatter
plot**

80_day_charge

80_total_minutes

80_total_calls



Using Prepare Recipe for Data Cleaning

Eliminating Outliers 18 steps

605 — 128

- Clear values outside [6.200,55.08] in 80_Total day charge
49
- Keep cells in 80_Account length only if $1 \leq \text{value} \leq 208$
48
- Clear values outside [46.50,154.5] in 80_Total day calls
46
- Clear values outside [58.55,342.2] in 80_Total eve minutes
56
- Keep rows where $46.75 \leq 80_Total \text{ eve calls} \leq 146.25$
— 128

Removing Outliers

Filling empty rows 13 steps

605

- Fill empty cells of 80_Total day charge with '0.0'
49
- Fill empty cells of 80_Account length with '0'
48
- Fill empty cells of 80_Total day calls with '0'
46
- Fill empty cells of 80_Total eve minutes with '0'
52
- Fill empty cells of 80_Total intl charge with '0.0'
20

Filling Empty Rows

Round values in 80_total minutes

9872

- Rename 42 columns
9872
- Move 80_day_charge before 80_day_calls

Renaming Columns

- Create column revenue_loss with formula
2601

Output column
revenue_loss

Expression
`if(prediction == 1, (80_day_charge + 80_eve_charge + 80_night_charge +`

Feature Engineering

Using Python Recipe for K-Means Clustering Model building

```
import dataiku
import pandas as pd, numpy as np
from dataiku import pandasutils as pdu
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

ip_dataset = dataiku.Dataset("churn_bigml_80_joined_prepared_prepared_filtered_cols")
ip_df = ip_dataset.get_dataframe()
n = 3
clustering_df = ip_df.copy()

scaler = StandardScaler()
scaled_clustering_data = scaler.fit_transform(clustering_df)
scaled_clustering_df = pd.DataFrame(scaled_clustering_data, columns = clustering_df.columns, index = clustering_df.index)

kmeans = KMeans(n_clusters = n, random_state = 42, n_init = 10)
kmeans.fit(scaled_clustering_df)

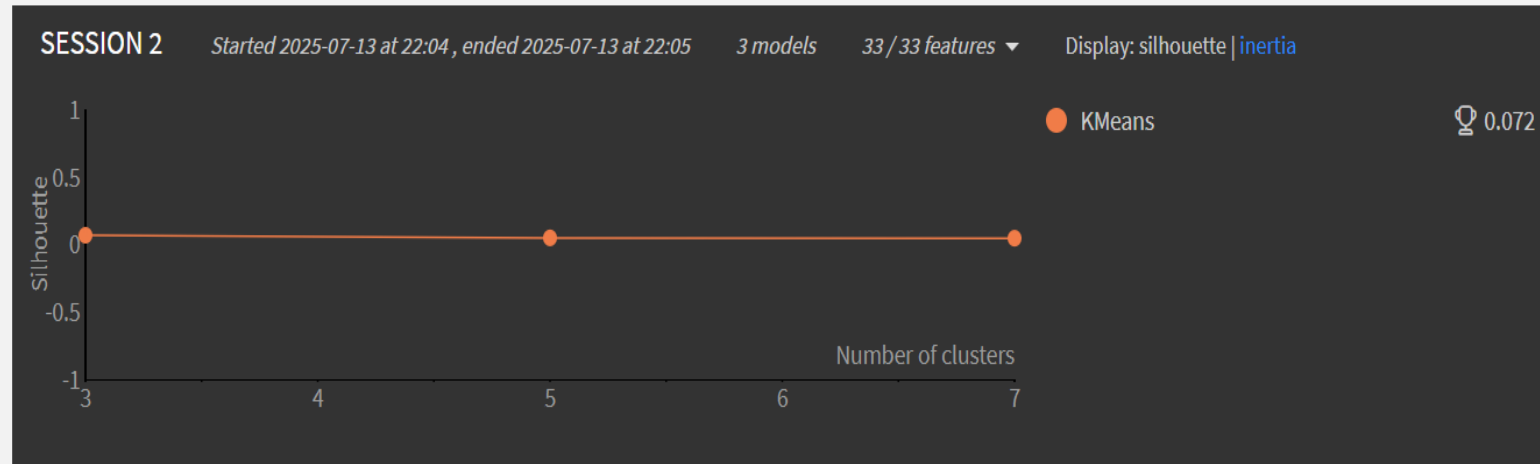
cluster_labels = pd.Series(kmeans.labels_, index=scaled_clustering_df.index)
op_df = ip_df.copy()
op_df['cluster_id'] = cluster_labels.reindex(op_df.index)

op_dataset = dataiku.Dataset("Kmeans")
op_dataset.write_with_schema(op_df)
```


K-MEANS CLUSTERING

Previously trained

<input type="checkbox"/>	SESSION 2		
<input checked="" type="checkbox"/>	KMeans (k=3) (s2)	0.072	★
<input type="checkbox"/>	KMeans (k=5) (s2)	0.052	★
<input type="checkbox"/>	KMeans (k=7) (s2)	0.050	★



KMeans (k=3)

Cluster outliers

Trained in 30 seconds on 13009 records

cluster_0	<div><div></div></div>	5511 (42.36%)
cluster_1	<div><div></div></div>	5221 (40.13%)
cluster_2	<div><div></div></div>	2277 (17.50%)

- The model with **k=3** has the highest silhouette score (**0.072**), indicating it's the best clustering option among the three tested.

Observations

- **80_total_minutes** is in average **12.28% greater** : mean of 652 against 581 globally
- **80_eve_minutes** is in average **13.34% greater** : mean of 226 against 200 globally
- **80_eve_charge** is in average **13.36% greater** : mean of 19.23 against 16.96 globally

The observations indicate heavy usage customers