

**INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT AKURDI, PUNE**

Documentation On

**“**PREDICTIVE ANALYSIS OF TELECOM CHURN FOR OPTIMAL CUSTOMER RETENTION**”**

PG-DBDA March 2023

**Mr. Rohit Puranik**

**Centre Coordinator**

**Mr. Abhijit Nagargoje**

**Project Guide**

**Submitted by-**

Group no.4

Piyush khichi (233530)

Chaitanya Patil (23350)

# **Table of Contents**

[**Table of Contents** 1](#_Toc144213011)

[**Abstract** 3](#_Toc144213012)

[**Data Collection and Pre processing** 4](#_Toc144213013)

[**Crawler** 4](#_Toc144213014)

[**Train/ Test split** 5](#_Toc144213015)

[**Face Detection** 6](#_Toc144213016)

[**Haar Caascade** 6](#_Toc144213017)

[**MTCNN** 10](#_Toc144213018)

[**Haar cascade vs MTCNN** 12](#_Toc144213019)

[**Machine Learning Models** 14](#_Toc144213020)

[**FaceNet** 14](#_Toc144213021)

[**Inception ResNet v2** 16](#_Toc144213022)

[**VGG16** 18](#_Toc144213023)

[**FaceNet vs VGG16** 21](#_Toc144213024)

[**Machine Learning Frameworks** 23](#_Toc144213025)

[**Tensorflow** 23](#_Toc144213026)

[**Keras** 24](#_Toc144213027)

[**Transfer Learning** 25](#_Toc144213028)

[**Softmax classifier** 26](#_Toc144213029)

[**L1 vs L2 Losses** 29](#_Toc144213030)

[**Pickle vs HDF5** 30](#_Toc144213031)

[**Face Recognition** 31](#_Toc144213032)

[**OpenCV** 33](#_Toc144213033)

[**Project Source codes** 35](#_Toc144213034)

[**Phase 1 : Data Collection and Pre Processing** 35](#_Toc144213035)

[**Phase 2 : Feature Extration / Encodings Generation** 37](#_Toc144213036)

[**Phase 3 : Face Recognition** 41](#_Toc144213037)

[**Prediction screenshots of MTCNN + FaceNet model:** 46](#_Toc144213038)

[**Prediction screenshots of VGG16 model:** 47](#_Toc144213039)

[**Conclusion** 48](#_Toc144213040)

[**Future Scope** 48](#_Toc144213041)

[**References** 50](#_Toc144213042)

Table of Figures

[Figure 1 Crawler Architecture 5](#_Toc144156569)

[Figure 2 Types of Cascading Functions 7](#_Toc144156570)

[Figure 3 Example of Haar Cascade 9](#_Toc144156571)

[Figure 4 Flow of MTCNN 11](#_Toc144156572)

[Figure 5 Haar Cascade vs MTCNN 13](#_Toc144156573)

[Figure 6 Comparison of face detected by both model 14](#_Toc144156574)

[Figure 7 Traditional ML vs Transfer Learning 18](#_Toc144156575)

[Figure 8 Flow of FaceNet 18](#_Toc144156576)

[Figure 9 FaceNet architecture 19](#_Toc144156577)

[Figure 10 Cost Function 19](#_Toc144156578)

[Figure 11 Triplet-loss and learning 19](#_Toc144156579)

[Figure 12 Architecture of Inception ResNet v2 21](#_Toc144156580)

[Figure 13 Softmax activation function 22](#_Toc144156581)

[Figure 14 Graph of softmax function 23](#_Toc144156582)

[Figure 15 L1 and L2 Loss Formiula 24](#_Toc144156583)

[Figure 16 Architecture of VGG16 26](#_Toc144156584)

# **Abstract**

# Telecommunication plays a major role In day to day life. With increase in demand in the telecom industry and there is though competition going among the telecom companies. Customer churn is widely spread problem among the telecom industry. Customer churn has huge impact on profits of the company. It will be great use for the companies to predict the customer churn over a particular period of time. So that they do something that make the customer not to change the network A study shows that there is increase in churn rate in the telecom Companies. By using Machine Learning Algorithms We can perform predictions on the data. In prediction of customer churn rate we are using Support Vector Machine(SVM) and Gradient Boosting (XGBoost) Algorithms. Results show that Extra Trees Classifier, XGBoosting Algorithm and Support Vector Machine have the best churn modelling performance, particularly for 80:20 dataset distribution.

# **INTRODUCTION**

# Telecom industry is constantly evolving and innovating. Due to the ever-increasing competition among the corporations, an increased importance is being given to targeted promoting methods for customer churn management. Modern day customers expect the best services at affordable rates. In case they are not satisfied, they quickly switch to another telecom network. Companies must find innovative ways to predict potential customer churn in order to prosper in such a competitive market. Customer churn is defined as the proportion of customers who stopped using a particular company’s products or services during a definite time frame. Mathematically.

**2.1.1 Basics of ML:**

Machine Learning is an Application of Artificial Intelligence (AI) it gives devices the ability to learn from their experiences and improve their self without doing any coding

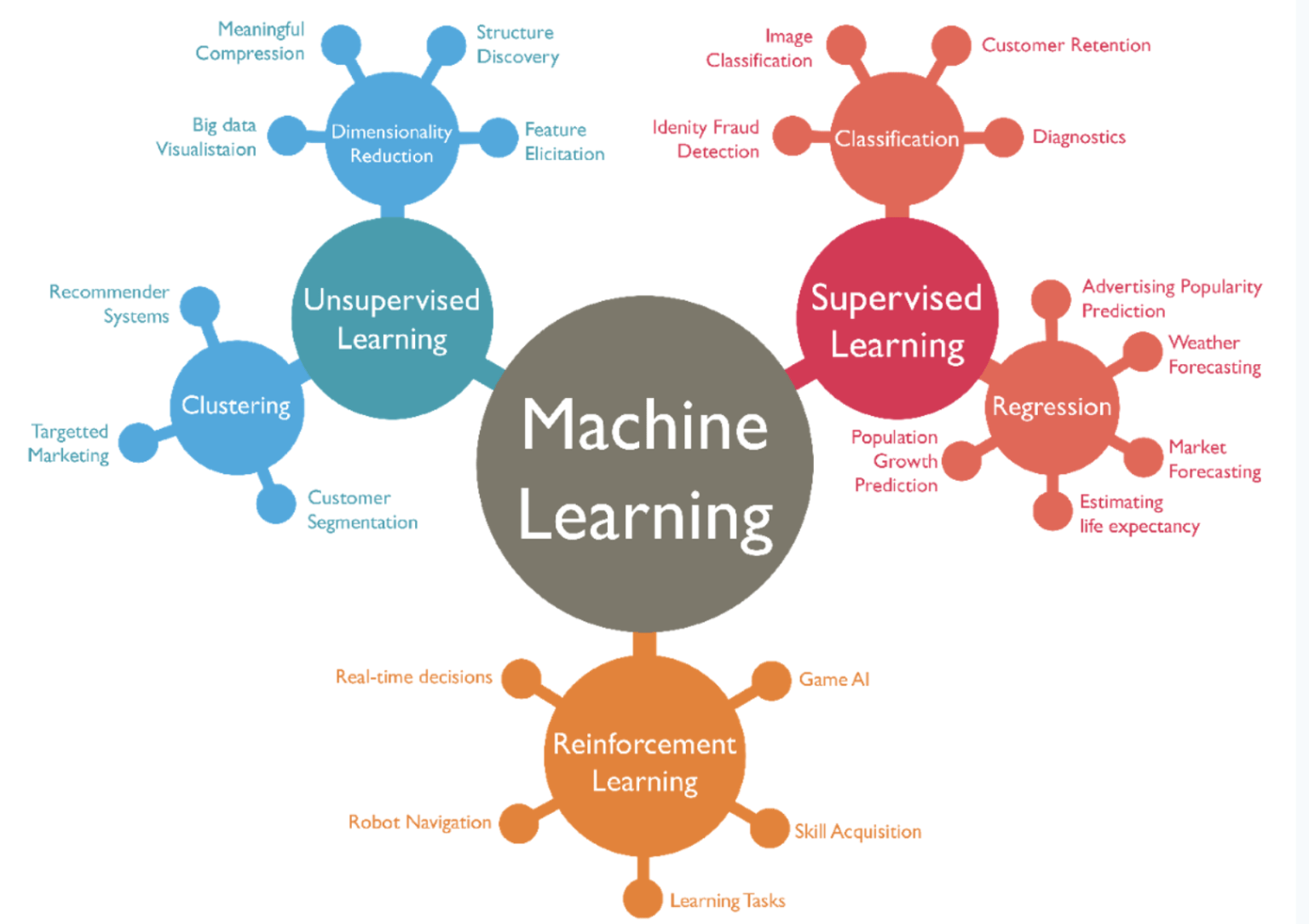


Figure 2.1.1: Outline of ML

Machine learning is the science of getting computers to act without being explicitly programmed. In the past decade, machine learning has given us self-driving cars, practical speech recognition, effective web search, and a vastly improved understanding of the human genome. Machine learning is so pervasive today that you probably use it dozens of times a day without knowing it.

**2.1.2 How does ML works?**

At its heart, machine learning algorithms analyse and identify patterns from datasets and use this information to make better predictions on new data sets.It is similar to how humans learn and improve. Whenever we make a decision, we consider our past experiences to assess the situation better. A machine learning model does the same by analysing historical data to make predictions or decisions. After all, machine learning is an AI application that enables machines to self-learn from data.

****

Fig.2.1.2 BLOCK DIAGRAM OF ML

**2.1.3 Features of ML**

The most important features of ML on which it works are Automation, Data Analysis Some of them are listed below:

• Automated Data Visualization

• Accurate Data Analysis

• Business Intelligence

• Artificial Intelligence

• Predictions

• Customer Engagemen

**2.1.4 Advantages**

• Efficient Handling of Data

• Automation For Everything

• Wide range of applications

• Continuous Improvement

• Easy Identification of Trends and Patterns

**2.1.5 Disadvantages**

• Data Acquisition

• Time Consuming

• Result Interpretation

**2.1.6 Python:**

Python is a high-level, general-purpose and a very popular programming language. Python programming language (latest Python 3) is being used in web development, Machine Learning applications, along with all cutting edge technology in Software Industry. modules and packages, which encourages program modularity and code reuse. The python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed. Python has number of libraries the following are the libraries that we have used:

**Pandas:**

Python Pandas is defined as an open-source library that provides high-performance data manipulation in Python.The name of Pandas is derived from the word panel data, which means an ecometrics from Multidimensional data.It is used for data analysis in Python and developed by wes mckineey in 2008. Data analysis requires lots of processing, such as restricting,cleaning, or merging etc. There are different tools are available for fast data processing, such asNumpy, Scipy and pandas. But we prefer Pandas because working with Pandas is fast, simple and more expressive than other tools. Pandas is built on top of the Numpy package, means Numpy, is required for operating the Pandas.

**Numpy:**

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

**Seaborn:**

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated to the data structures from Pandas. Seaborn aims to make visualization the central part of exploring and understanding data.

**Matplotlib:**

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

**2.1.7 Jupyter Notebook:**

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. Its uses include data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and much more

**2.2 About The Project: (Telecom Churn Prediction)**

Telecommunication is playing a major role in our day to day life. By using Mobiles, Mails we can communicate to people who are in various places and there is a lot of competition among the telecom companies. The main problem for the telecom companies is that the telecom churn. Due to various factors like signals, raise in cost customers are changing to another network. There will be loss to companies if they are loosing the customers. Predictions will be very helpful to the telecom companies, By using Machine Learning Algorithms like Support Vector Machine and XGBoost we can predicate the churn rate of the given customers. The companies can provide some benefits to the customers who are going to churn and they have a less chance of loosing a customer.

**3. PROBLEM STATEMENT**

Customer Churn is major problem going on in many companies. Customer Churn in the telecom industry means moving of customers from one network to the other network . Loosing of a customer has a so much impact on the company. There are many reasons that a customer want to change the network for examples they can find better recharge plans in other network or there are offering many offers or they are changing of places. If we can predict customer churn effectively and accurately then the telecom companies can provide better services for them. The Churn Prediction model can be done based on the machine learning techniques like Support Vector Machine(SVM) and XGBoost. The efficiency of these algorithms is more as compared to other algorithms and result can be predicted more accurately.

**4. EXSISTING SYSTEM**

In existing System, the telecom churn prediction is done through using the algorithms of machine learning and in python. The algorithms that are used in existing system to predict the churning in the telecom industry are Random forest, Adaboost, KNN Classifiers and other algorithms are used to predict the churn of the customers. The algorithms that are used in existing will give the low accuracy and due to the low accuracy some prediction may not be correct always.

**Algorithms Used:**

• Logistic Regression

• Random Forest

• AdaBoost

• Decision Trees

**Drawbacks:**

If the efficiency given by the algorithm is low then the prediction will not be accurate and thereby it can lead to loose of customers for telecom companies.

**5. PROPOSED SYSTEM**

In the proposes System, We are taking the data of the customers from the past in the form a dataset and based on the characteristics like their previous recharges and some other attributes. For predicting the customer churn we are using machine learning with python. In the dataset that we have taken consists of data about 1000 members which is collected from Kaggle. Here we are using Supervised Machine Learning Algorithms like Support Vector Machine (SVM) and XGBoost we are predicting the accurate values of the churn.

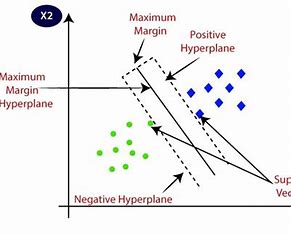
**5.1 Algorithms Used:**

• Support Vector Machine (SVM)

• Extreme Gradient Boosting (XGBoost)

**5.1.1.Support Vector Machine (SVM):**

Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression. SVMs have their unique way of implementation as compared to other machine learning algorithms. Lately, they are extremely popular because of their ability to handle multiple continuous and categorical variables.



**Fig.5.1.1.Support Vector Machine**

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyper plane. SVM chooses the extreme points/vectors that help in creating the hyper plane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

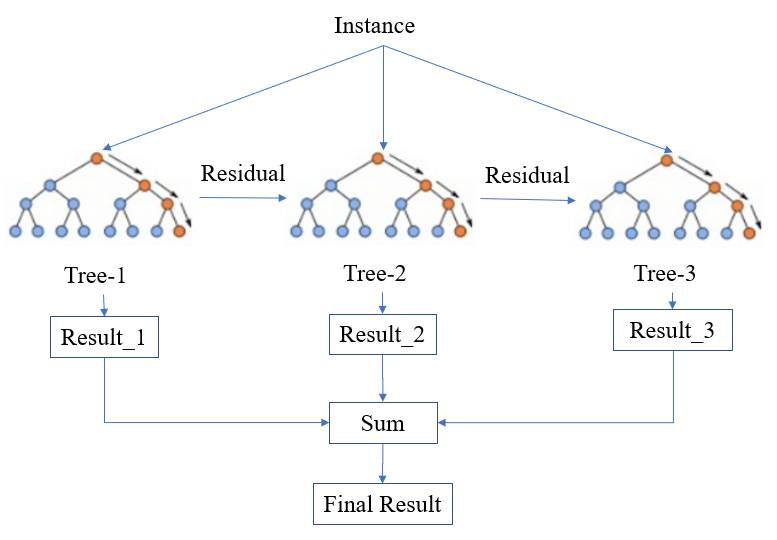
**5.1.2 Extreme Gradient Boosting (XGBoost):**

XGBoost classifier is a Machine Learning Algorithm that is applied for structured and tabular data. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. XGBoost works with large, complicated datasets. XGBoost is an ensemble modelling technique.



**5.1.2.XGBoost**

XGBoost is an ensemble learning method. Sometimes, it may not be sufficient to rely upon the results of just one machine learning model. Ensemble learning offers a systematic solution to combine the predictive power of multiple learners. In the case of boosting, the decision tree followed a sequential chain for learning. Each split sub-parts gets trained from its forerunner, and any kind of error existing in the current part gets rectified and leads to the next sub-part.



**5.1.2 XGBoost**

**Benefits:**

* The telecom Companies Can the churn rate of the customers.
* By predicting the churn rate the telecom industry can reduce the churn rate by providing better services
* By knowing the churn rate the companies can improve their services an thereby the risk of loosing a customer is less

**5.1.3 Steps for Applying Algorithm:**

**Step 1:** Defining Problem

**Step 2:** Collection Of Data

**Step 3:** Exploratory Data Analysis

**Step 4:** Train or Test Split the data

**Step 5:** Evaluate the Data

**Step 6:** Based on the dataset predict the analysis

**Step 7:** Displays the Accurate Result

**5.2 Components Used:**

**5.2.1 Software Requirements:**

Language Used : Python 3.10.0 Operating System :

Windows / Linux

Tools Used:

• Anaconda Navigator

• Jupyter Notebook

♣ Numpy

♣ Pandas

♣ Seaborn Matplotlib

**5.2.2 Hardware Requirements:**

Processor Intel corei3 or higher

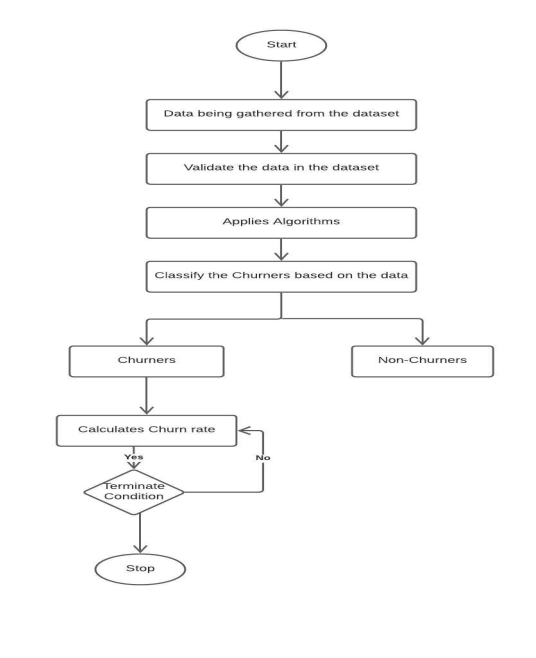
RAM 4GB or higher

**5.3.UML Diagrams:**

UML stands for Unified Modelling Language which is a standardized modelling language, consisting of integrated set of diagrams. UML diagrams are used to visualize a model of the system. These diagrams describe only the design or structure of the projects or system.

**5.3.1 Use Case Diagram:**

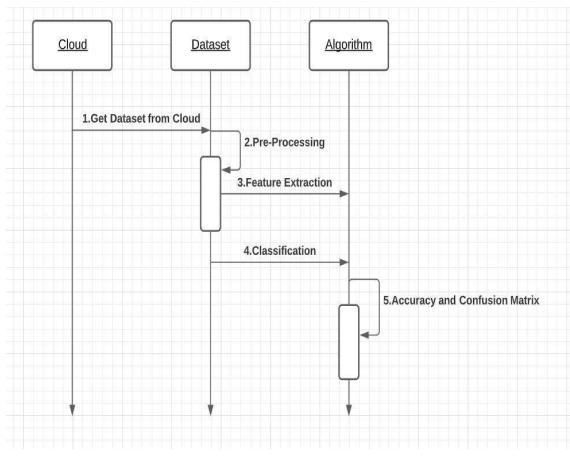
A Use case diagram describes a system’s functional requirements in terms of use-cases. It is a model of system’s use-cases and its actors. Here use-cases are similar to the actions performed. The actors of our system are User and Device.



**Figure 5.3.1: Use case Diagram for Telecom Churn Prediction**

**5.3.4 Sequence Diagram:**

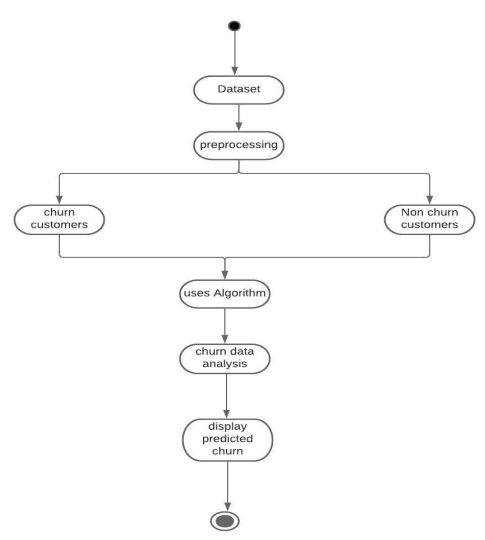
A Sequence diagram shows how the objects interact with others in a particular scenario of a use-case. This diagram describes the flow of events of a particular system



**Figure 5.3.5: Sequence Diagram for Telecom Churn Prediction**

**5.3.5 Activity Diagram:**

A Deployment diagram helps to model the physical aspects of a system. This diagram shows the architecture of the system. This diagrams involves modeling the hardware configurations together with the software components



**Figure 5.3.6: Activity Diagram for Telecom Churn Prediction**

**6.METHODOLOGY**

The basic layer for predicting future customer churn is data from the past. We look at data from customers that already have churned (response) and their characteristics / behaviour (predictors) before the churn happened. By fitting a statistical model that relates the predictors to the response, the response for existing customers is predicted. The overall scope of work to forecast customer attrition may look like the following

Understanding a problem and final goal

• Data collection

• Data preparation and pre-processing

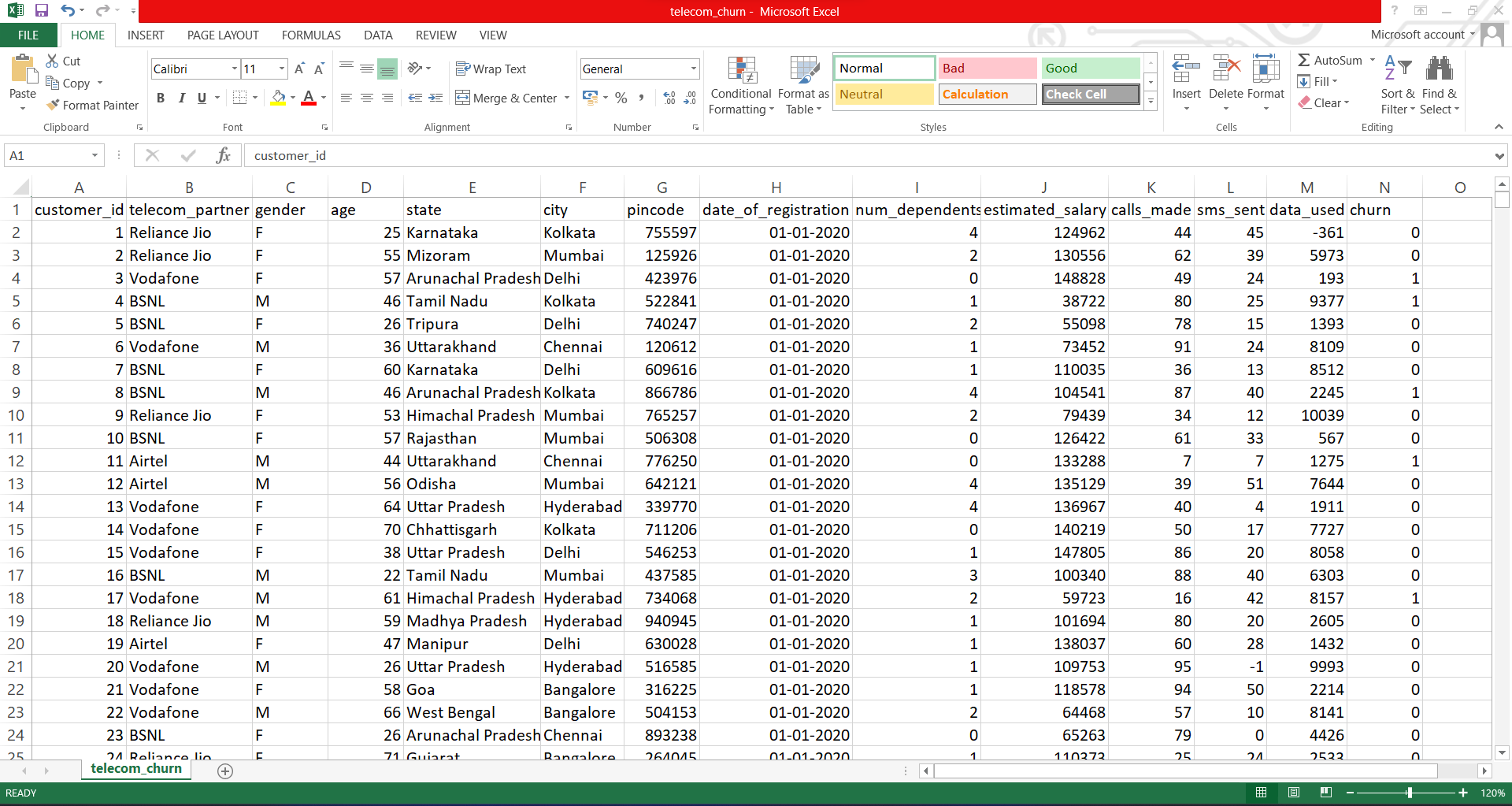
• Modelling and testing

• Model deployment and monitoring

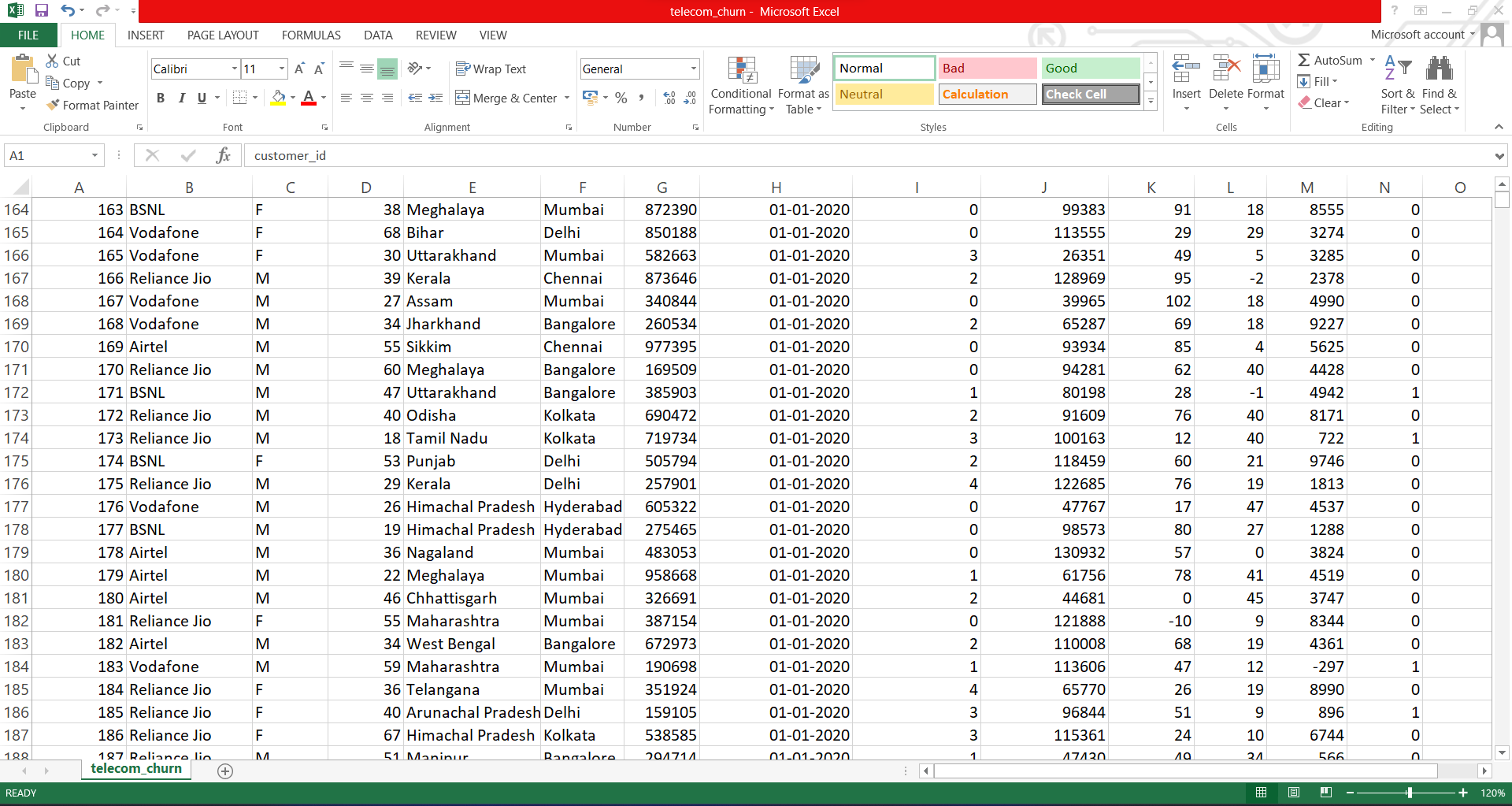
It’s important to understand what insights one needs to get from the analysis. In short, you must decide what question to ask and consequently what type of machine learning problem to solve: classification or regression.

**7. Telecom Dataset**

* Dataset Format = .csv
* No. of Rows = 243553
* No. of Columns = 14

****

**Fig.7.1 Trained Data**



**Fig.7.2 Trained Data**

**8. SOURCE CODE AND RESULT**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.metrics import accuracy\_score, classification\_report

from sklearn.metrics import f1\_score

from sklearn.metrics import confusion\_matrix

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.linear\_model import LogisticRegression

from xgboost import XGBClassifier

import xgboost as xgb

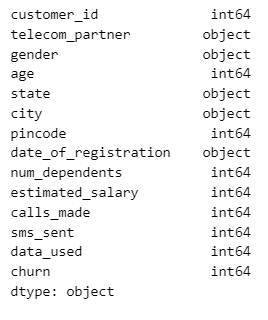
from catboost import CatBoostClassifier

import lightgbm as lgb

**# Load the dataset**

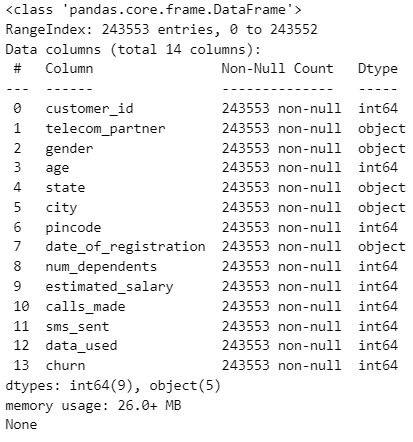
dataset = pd.read\_csv('Churning\_data.csv')

dataset.dtypes



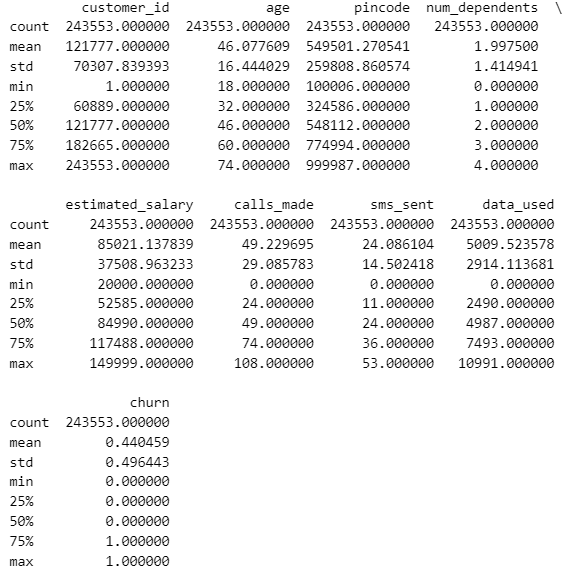
# Display basic information about the dataset

print(dataset.info())



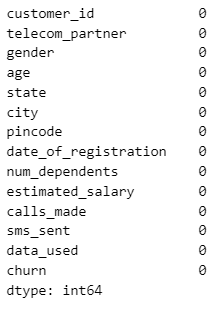
**# Summary statistics**

print(dataset.describe())



**# Check for missing values**

print(dataset.isnull().sum())

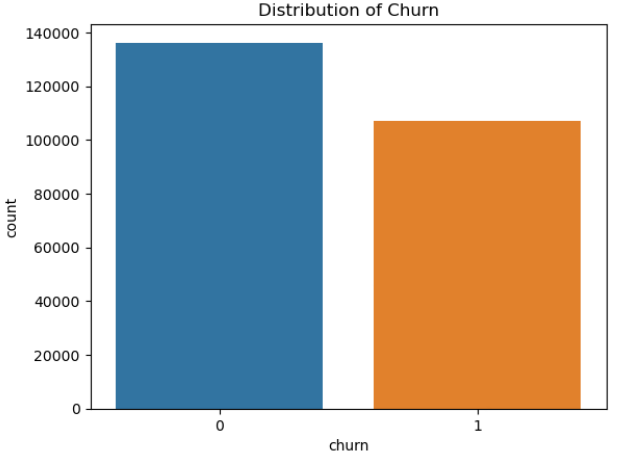


**# Check the distribution of the target variable**

sns.countplot(x='churn', data=dataset)

plt.title("Distribution of Churn")

plt.show()



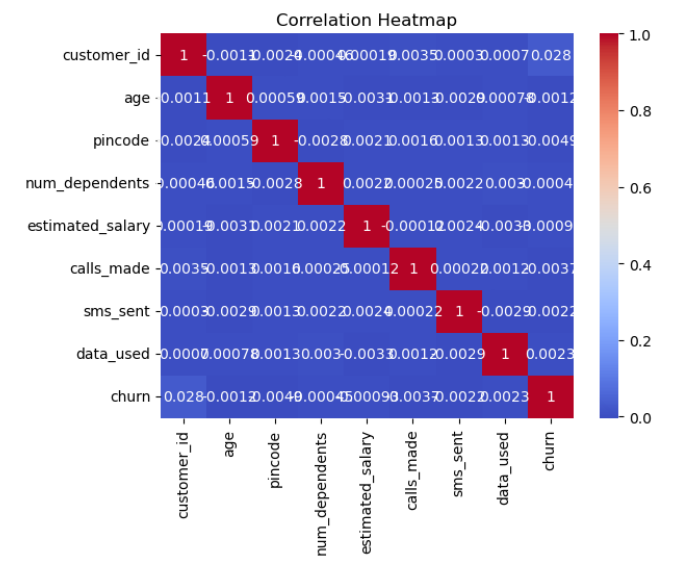
**# Visualize the correlation between numerical features**

corr\_matrix = dataset.corr()

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm')

plt.title("Correlation Heatmap")

plt.show()



**# Plot distribution of numerical features**

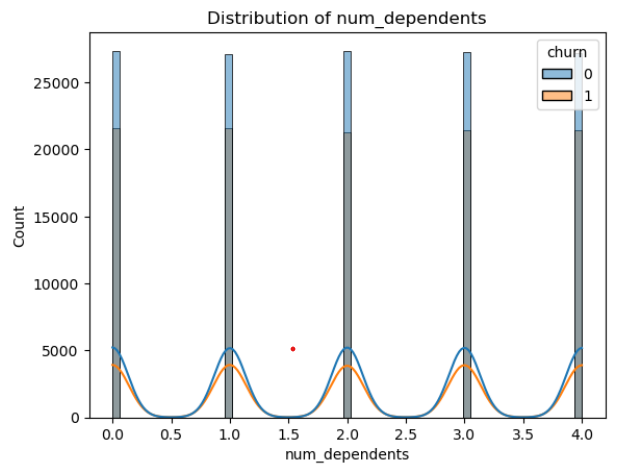
numerical\_features = ['num\_dependents', 'estimated\_salary', 'calls\_made', 'sms\_sent', 'data\_used', 'age']

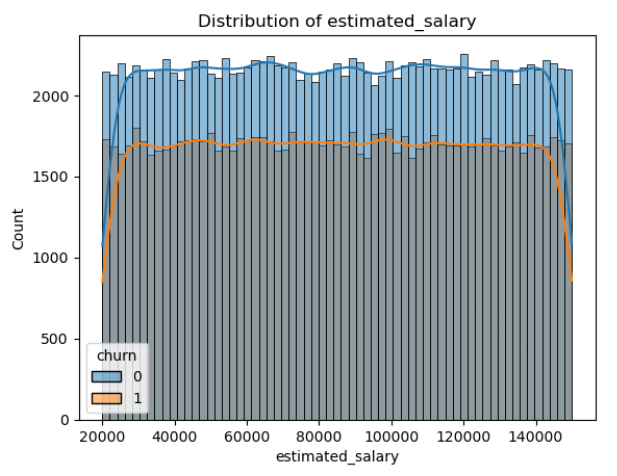
for feature in numerical\_features:

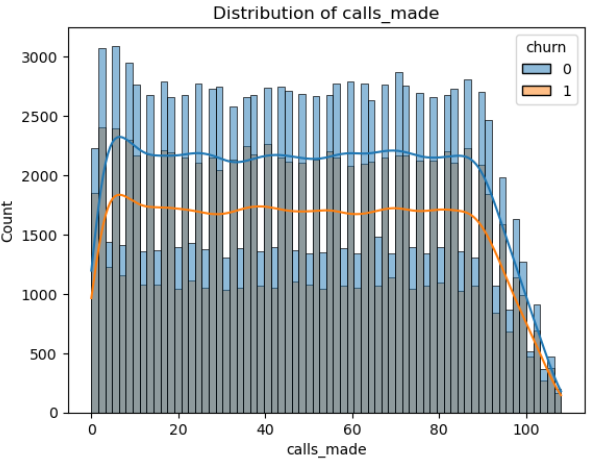
sns.histplot(data=dataset, x=feature, hue='churn', kde=True)

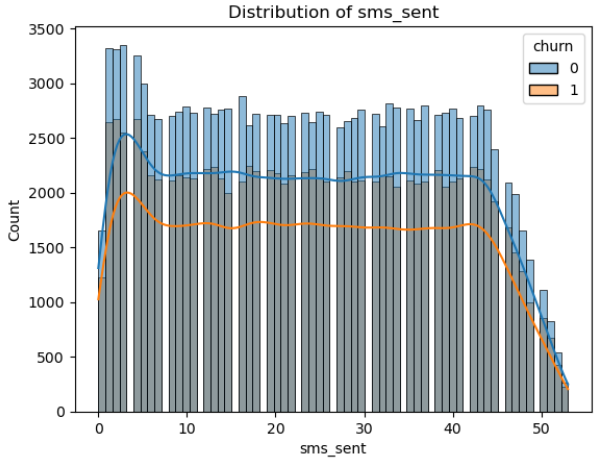
plt.title(f"Distribution of {feature}")

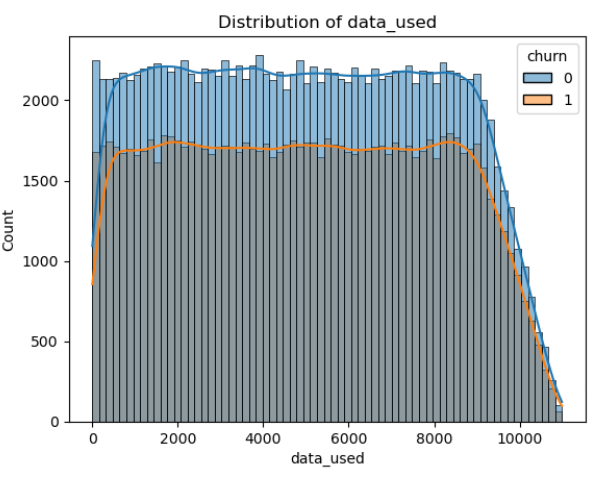
plt.show()

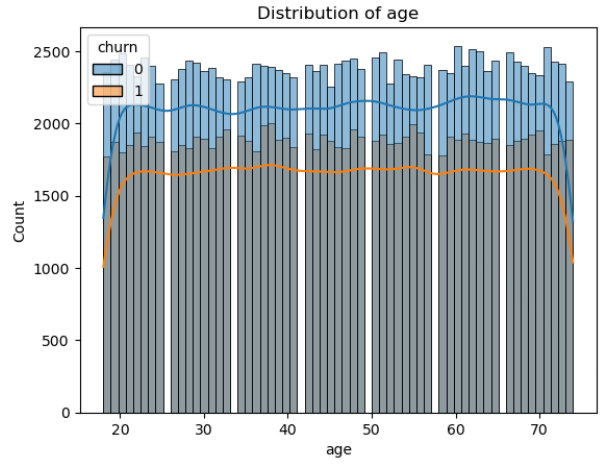












**# Analyze categorical features**

categorical\_features = ['telecom\_partner', 'gender', 'state','city']

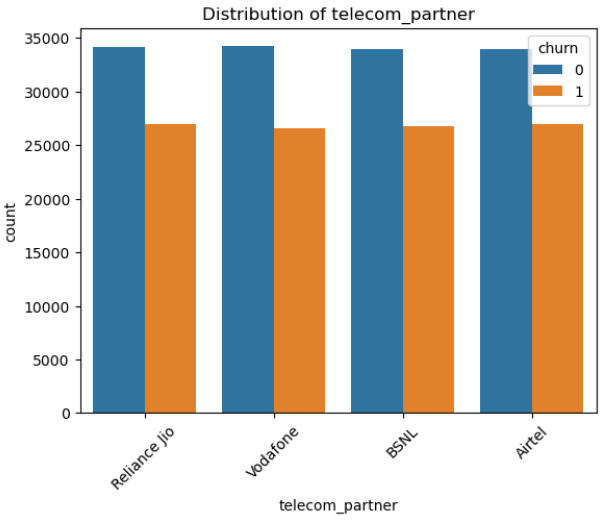
for feature in categorical\_features:

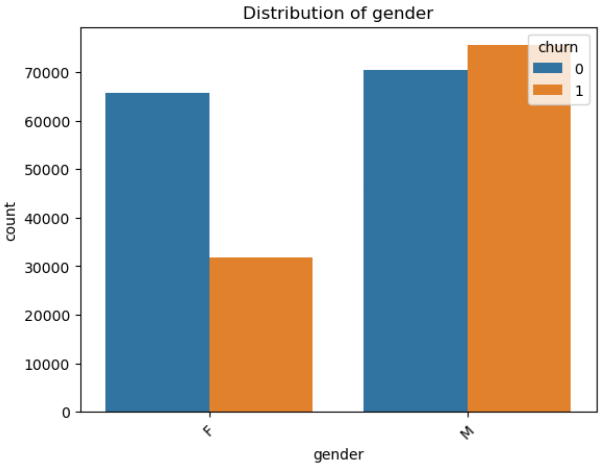
sns.countplot(x=feature, data=dataset, hue='churn')

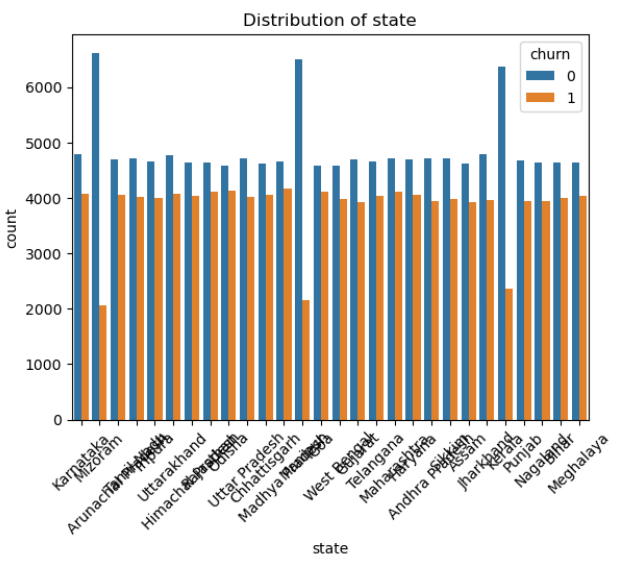
plt.title(f"Distribution of {feature}")

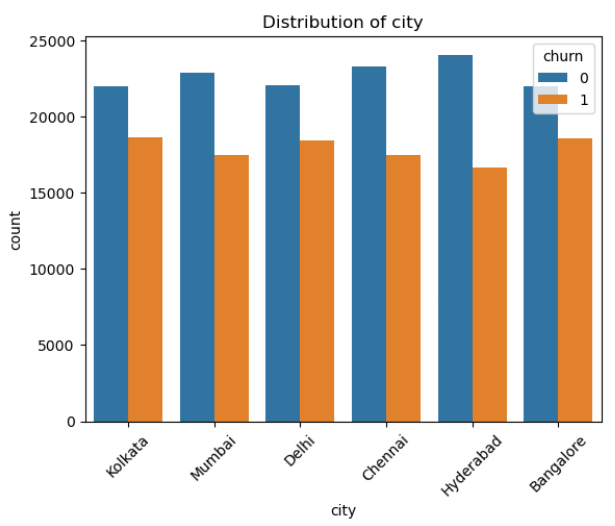
plt.xticks(rotation=45)

plt.show()









import pandas as pd

**# Assuming you have a DataFrame named df**

**# Replace 'DateColumn' with the actual name of the column containing the date values**

value\_to\_remove = '18-08-2021'

dataset = dataset[dataset['date\_of\_registration'] != value\_to\_remove]

**# Now df does not contain rows with the value '18-08-2021'**

**# List of categorical columns**

categorical\_columns = ['telecom\_partner', 'gender', 'state', 'city']

**# Perform one-hot encoding**

data\_encoded = pd.get\_dummies(dataset, columns=categorical\_columns)

**# Split the data into training and testing sets**

X = data\_encoded.drop('churn', axis=1)

y = data\_encoded['churn']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42, shuffle=True)

**# Create an instance of StandardScaler**

scaler = StandardScaler()

**# Drop the date column before scaling**

X\_train\_without\_date = X\_train.drop('date\_of\_registration', axis=1)

X\_test\_without\_date = X\_test.drop('date\_of\_registration', axis=1)

**# Fit the scaler on the training data and transform both training and test data**

X\_train\_scaled = scaler.fit\_transform(X\_train\_without\_date)

X\_test\_scaled = scaler.transform(X\_test\_without\_date)

from sklearn.tree import DecisionTreeClassifier

from sklearn.feature\_selection import SelectFromModel

sfm = SelectFromModel(DecisionTreeClassifier(), threshold="mean")

sfm.fit(X\_train\_scaled, y\_train)

X\_train\_sfm = sfm.transform(X\_train\_scaled)

X\_test\_sfm = sfm.transform(X\_test\_scaled)

**# Get the selected feature indices**

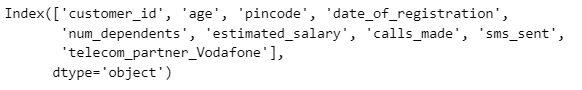
selected\_feature = sfm.get\_support(indices=True)

**# Get the original column names of the selected features**

selected\_feature\_names = X\_train.columns[selected\_feature]

**# Print the selected feature names**

print(selected\_feature\_names)



from sklearn.feature\_selection import RFE

**# Create an instance of the model you want to use for feature selection**

**# For example, let's use a RandomForestClassifier for this demonstration**

model = RandomForestClassifier()

**# Create an instance of RFE with the model and desired number of features to select**

num\_features\_to\_select = 6

**# Adjust this as needed**

rfe = RFE(model, n\_features\_to\_select=num\_features\_to\_select)

**# Fit RFE on the training data**

rfe.fit(X\_train\_scaled, y\_train)

**# Get the selected feature indices**

selected\_feature\_indices = rfe.support\_

**# Get the selected feature names**

selected\_feature\_names=X\_train\_without\_date.columns[selected\_feature\_indices]

**# Print the selected feature names**

print("Selected features:")

for feature in selected\_feature\_names:

print(feature)

**# Apply Logistic Regression**

print("Logistic Regression:")

log\_reg = LogisticRegression(class\_weight='balanced', C=0.001, penalty='l2', random\_state=42)

log\_reg.fit(X\_train\_scaled, y\_train)

log\_reg\_y\_pred = log\_reg.predict(X\_test\_scaled)

log\_reg\_accuracy = accuracy\_score(y\_test, log\_reg\_y\_pred)

log\_reg\_f1 = f1\_score(y\_test, log\_reg\_y\_pred)

log\_reg\_conf\_matrix = confusion\_matrix(y\_test, log\_reg\_y\_pred)

log\_reg\_report = classification\_report(y\_test, log\_reg\_y\_pred, target\_names=['Not Churn', 'Churn'])

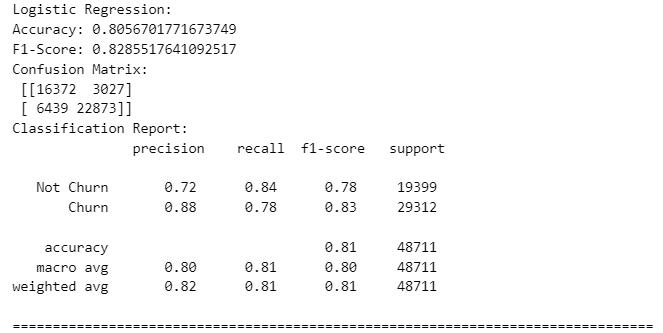
print("Accuracy:", log\_reg\_accuracy)

print("F1-Score:", log\_reg\_f1)

print("Confusion Matrix:\n", log\_reg\_conf\_matrix)

print("Classification Report:\n", log\_reg\_report)

print("=" \* 80)



from sklearn.model\_selection import GridSearchCV

from sklearn.linear\_model import LogisticRegression

# Define hyperparameter grid for fine-tuning

param\_grid = {

'C': [0.001, 0.01, 0.1, 1, 10],  **# Regularization parameter**

'penalty': ['l1', 'l2'],          **# Regularization type**

'class\_weight': [None, 'balanced']  **# Class weight**

}

**# Initialize the Logistic Regression model**

log\_reg = LogisticRegression(random\_state=42)

**# Create GridSearchCV instance**

grid\_search = GridSearchCV(log\_reg, param\_grid, scoring='f1', cv=5, n\_jobs=-1)

**# Fit the GridSearchCV on training data**

grid\_search.fit(X\_train\_scaled, y\_train)

**# Get the best estimator**

best\_log\_reg = grid\_search.best\_estimator\_

**# Predict using the best model**

log\_reg\_y\_pred = best\_log\_reg.predict(X\_test\_scaled)

**# Calculate performance metrics**

log\_reg\_accuracy = accuracy\_score(y\_test, log\_reg\_y\_pred)

log\_reg\_f1 = f1\_score(y\_test, log\_reg\_y\_pred)

log\_reg\_conf\_matrix = confusion\_matrix(y\_test, log\_reg\_y\_pred)

log\_reg\_report = classification\_report(y\_test, log\_reg\_y\_pred, target\_names=['Not Churn', 'Churn'])

print("Best Parameters:", grid\_search.best\_params\_)

print("Best F1-Score:", grid\_search.best\_score\_)

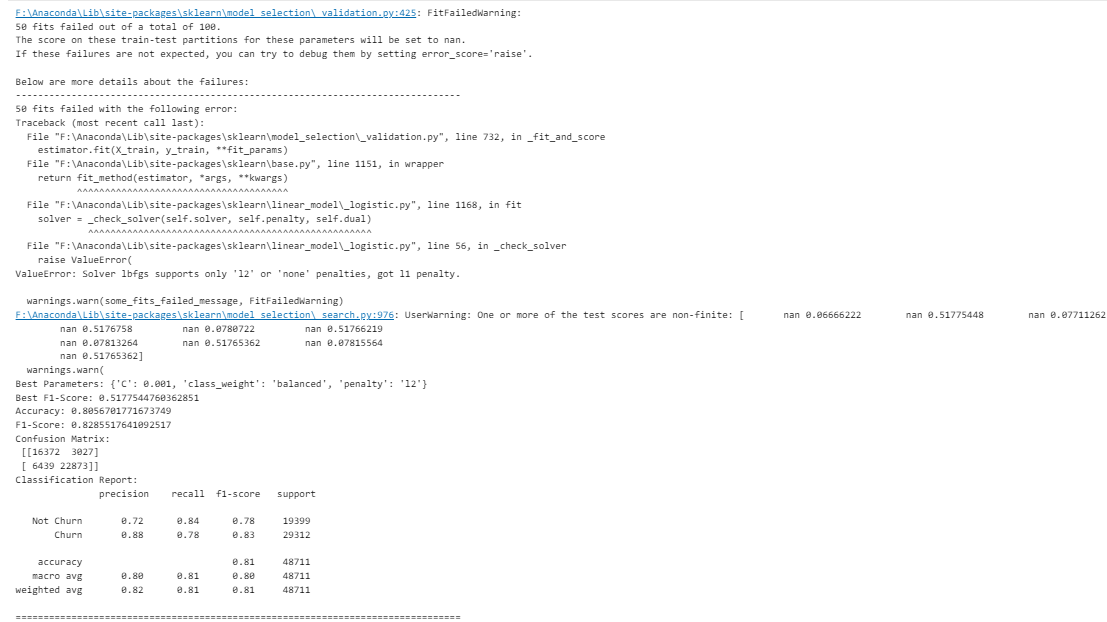
print("Accuracy:", log\_reg\_accuracy)

print("F1-Score:", log\_reg\_f1)

print("Confusion Matrix:\n", log\_reg\_conf\_matrix)

print("Classification Report:\n", log\_reg\_report)

print("=" \* 80)



**# Random Forest Classifier**

print("Random Forest Classifier:")

rf\_classifier = RandomForestClassifier(n\_estimators=500,

max\_depth=10,

min\_samples\_split=5,

min\_samples\_leaf=5,

max\_features='sqrt',

class\_weight='balanced',

random\_state=42)

rf\_classifier.fit(X\_train\_scaled, y\_train)

rf\_y\_pred = rf\_classifier.predict(X\_test\_scaled)

rf\_accuracy = accuracy\_score(y\_test, rf\_y\_pred)

rf\_f1 = f1\_score(y\_test, rf\_y\_pred)

rf\_conf\_matrix = confusion\_matrix(y\_test, rf\_y\_pred)

rf\_report = classification\_report(y\_test, rf\_y\_pred, target\_names=['Not Churn', 'Churn'])

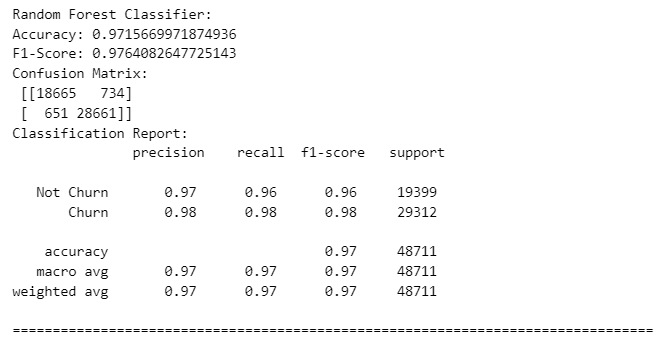
print("Accuracy:", rf\_accuracy)

print("F1-Score:", rf\_f1)

print("Confusion Matrix:\n", rf\_conf\_matrix)

print("Classification Report:\n", rf\_report)

print("=" \* 80)



from sklearn.model\_selection import GridSearchCV

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, f1\_score, confusion\_matrix, classification\_report

**# Define hyperparameter grid for fine-tuning**

param\_grid = {

'n\_estimators': [100, 200, 300, 500],

'max\_depth': [None, 10, 20, 30],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 5, 10],

'max\_features': ['auto', 'sqrt', 'log2'],

'class\_weight': [None, 'balanced']

}

**# Initialize the Random Forest Classifier model**

rf\_classifier = RandomForestClassifier(random\_state=42)

**# Create GridSearchCV instance**

grid\_search = GridSearchCV(rf\_classifier, param\_grid, scoring='f1', cv=5, n\_jobs=-1)

**# Fit the GridSearchCV on training data**

grid\_search.fit(X\_train\_scaled, y\_train)

**# Get the best estimator**

best\_rf\_classifier = grid\_search.best\_estimator\_

**# Predict using the best model**

rf\_y\_pred = best\_rf\_classifier.predict(X\_test\_scaled)

**# Calculate performance metrics**

rf\_accuracy = accuracy\_score(y\_test, rf\_y\_pred)

rf\_f1 = f1\_score(y\_test, rf\_y\_pred)

rf\_conf\_matrix = confusion\_matrix(y\_test, rf\_y\_pred)

rf\_report = classification\_report(y\_test, rf\_y\_pred, target\_names=['Not Churn', 'Churn'])

print("Best Parameters:", grid\_search.best\_params\_)

print("Best F1-Score:", grid\_search.best\_score\_)

print("Accuracy:", rf\_accuracy)

print("F1-Score:", rf\_f1)

print("Confusion Matrix:\n", rf\_conf\_matrix)

print("Classification Report:\n", rf\_report)

**# Apply Gradient Boosting**

print("Gradient Boosting:")

**# Create an instance of GradientBoostingClassifier**

gradient\_boosting = GradientBoostingClassifier(n\_estimators=100, random\_state=42)

**# Fit the model**

gradient\_boosting.fit(X\_train\_scaled, y\_train)

**# Predict using the model**

gradient\_boosting\_y\_pred = gradient\_boosting.predict(X\_test\_scaled)

**# Calculate metrics**

gradient\_boosting\_accuracy = accuracy\_score(y\_test, gradient\_boosting\_y\_pred)

gradient\_boosting\_f1 = f1\_score(y\_test, gradient\_boosting\_y\_pred)

gradient\_boosting\_conf\_matrix = confusion\_matrix(y\_test, gradient\_boosting\_y\_pred)

gradient\_boosting\_report = classification\_report(y\_test, gradient\_boosting\_y\_pred, target\_names=['Not Churn', 'Churn'])

**# Print metrics**

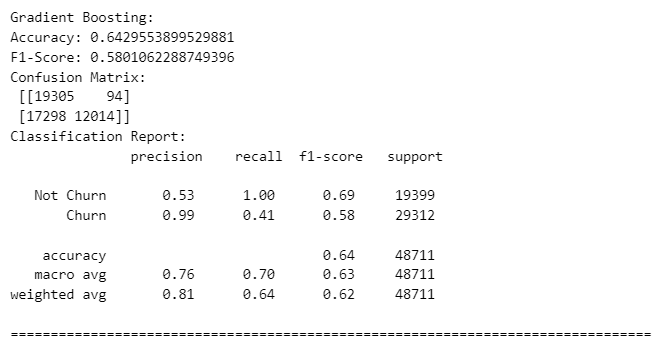
print("Accuracy:", gradient\_boosting\_accuracy)

print("F1-Score:", gradient\_boosting\_f1)

print("Confusion Matrix:\n", gradient\_boosting\_conf\_matrix)

print("Classification Report:\n", gradient\_boosting\_report)

print("=" \* 80)



**# Apply XGBoost Classifier**

print("XGBoost Classifier:")

xgb\_classifier = xgb.XGBClassifier(

n\_estimators=100,

max\_depth=6,

learning\_rate=0.1,

subsample=0.8,

colsample\_bytree=0.8,

random\_state=42

)

xgb\_classifier.fit(X\_train\_scaled, y\_train)

xgb\_y\_pred = xgb\_classifier.predict(X\_test\_scaled)

xgb\_accuracy = accuracy\_score(y\_test, xgb\_y\_pred)

xgb\_f1 = f1\_score(y\_test, xgb\_y\_pred)

xgb\_conf\_matrix = confusion\_matrix(y\_test, xgb\_y\_pred)

xgb\_report = classification\_report(y\_test, xgb\_y\_pred, target\_names=['Not Churn', 'Churn'])

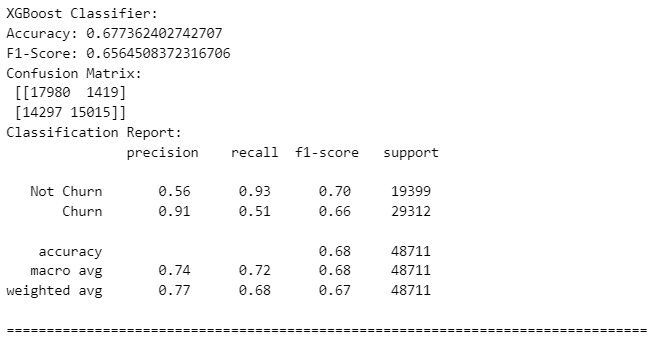
print("Accuracy:", xgb\_accuracy)

print("F1-Score:", xgb\_f1)

print("Confusion Matrix:\n", xgb\_conf\_matrix)

print("Classification Report:\n", xgb\_report)

print("=" \* 80)



from xgboost import XGBClassifier

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import accuracy\_score, f1\_score, confusion\_matrix, classification\_report

**# Define hyperparameter grid for fine-tuning**

param\_grid = {

'n\_estimators': [100, 200, 300],

'max\_depth': [6, 8, 10],

'learning\_rate': [0.01, 0.1, 0.2],

'subsample': [0.8, 1.0],

'colsample\_bytree': [0.8, 1.0]

}

**# Initialize the XGBoost Classifier**

xgb\_classifier = XGBClassifier(random\_state=42)

**# Create GridSearchCV instance**

grid\_search = GridSearchCV(xgb\_classifier, param\_grid, scoring='f1', cv=5, n\_jobs=-1)

**# Fit the GridSearchCV on training data**

grid\_search.fit(X\_train\_scaled, y\_train)

**# Get the best estimator**

best\_xgb\_classifier = grid\_search.best\_estimator\_

**# Predict using the best model**

xgb\_y\_pred = best\_xgb\_classifier.predict(X\_test\_scaled)

**# Calculate performance metrics**

xgb\_accuracy = accuracy\_score(y\_test, xgb\_y\_pred)

xgb\_f1 = f1\_score(y\_test, xgb\_y\_pred)

xgb\_conf\_matrix = confusion\_matrix(y\_test, xgb\_y\_pred)

xgb\_report = classification\_report(y\_test, xgb\_y\_pred, target\_names=['Not Churn', 'Churn'])

print("Best Parameters:", grid\_search.best\_params\_)

print("Best F1-Score:", grid\_search.best\_score\_)

print("Accuracy:", xgb\_accuracy)

print("F1-Score:", xgb\_f1)

print("Confusion Matrix:\n", xgb\_conf\_matrix)

print("Classification Report:\n", xgb\_report)

print("=" \* 80)

**# Create and train the LightGBM classifier**

lgb\_classifier = lgb.LGBMClassifier(

boosting\_type='gbdt',

num\_leaves=31,

max\_depth=-1,

learning\_rate=0.05,

n\_estimators=100,

class\_weight='balanced',

random\_state=42

)

lgb\_classifier.fit(X\_train\_scaled, y\_train)

**# Make predictions and evaluate the model**

lgb\_y\_pred = lgb\_classifier.predict(X\_test\_scaled)

lgb\_accuracy = accuracy\_score(y\_test, lgb\_y\_pred)

lgb\_f1 = f1\_score(y\_test, lgb\_y\_pred)

lgb\_conf\_matrix = confusion\_matrix(y\_test, lgb\_y\_pred)

lgb\_report = classification\_report(y\_test, lgb\_y\_pred, target\_names=['Not Churn', 'Churn'])

print("LightGBM Classifier:")

print("Accuracy:", lgb\_accuracy)

print("F1-Score:", lgb\_f1)

print("Confusion Matrix:\n", lgb\_conf\_matrix)

print("Classification Report:\n", lgb\_report)

print("=" \* 80)

**# Apply CatBoost Classifier**

print("CatBoost Classifier:")

catboost\_classifier = CatBoostClassifier(iterations=500,

depth=6,

learning\_rate=0.1,

loss\_function='Logloss',

verbose=100,

random\_seed=42)

catboost\_classifier.fit(X\_train\_scaled, y\_train)

catboost\_y\_pred = catboost\_classifier.predict(X\_test\_scaled)

catboost\_accuracy = accuracy\_score(y\_test, catboost\_y\_pred)

catboost\_f1 = f1\_score(y\_test, catboost\_y\_pred)

catboost\_conf\_matrix = confusion\_matrix(y\_test, catboost\_y\_pred)

catboost\_report = classification\_report(y\_test, catboost\_y\_pred, target\_names=['Not Churn', 'Churn'])

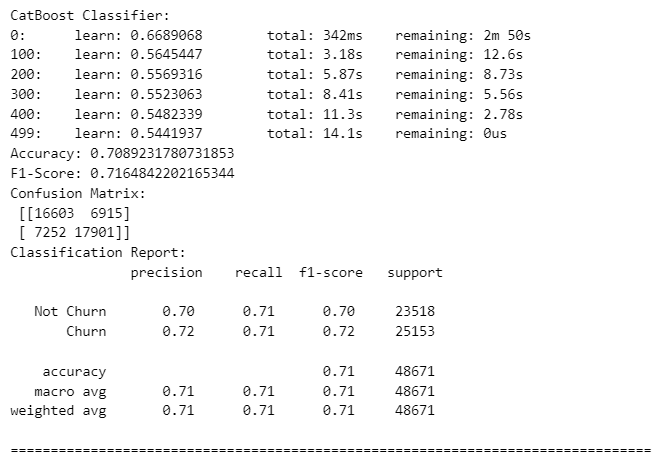
print("Accuracy:", catboost\_accuracy)

print("F1-Score:", catboost\_f1)

print("Confusion Matrix:\n", catboost\_conf\_matrix)

print("Classification Report:\n", catboost\_report)

print("=" \* 80)



from catboost import CatBoostClassifier, Pool

**# Define hyperparameter grid for fine-tuning**

param\_grid = {

'depth': [6, 8, 10],

'learning\_rate': [0.01, 0.1, 0.2],

'l2\_leaf\_reg': [1, 3, 5],

}

**# Initialize the CatBoostClassifier**

catboost\_classifier = CatBoostClassifier(iterations=1000, loss\_function='Logloss', verbose=100, random\_seed=42)

**# Create a Pool object for training**

train\_pool = Pool(X\_train\_scaled, y\_train)

**# Initialize CatBoost's GridSearch**

grid\_search = catboost\_classifier.grid\_search(param\_grid, train\_pool, cv=5, refit=True, partition\_random\_seed=42)

**# Get the best estimator**

best\_catboost\_classifier = grid\_search['params']

**# Predict using the best model**

catboost\_y\_pred = best\_catboost\_classifier.predict(X\_test\_scaled)

**# Calculate performance metrics**

catboost\_accuracy = accuracy\_score(y\_test, catboost\_y\_pred)

catboost\_f1 = f1\_score(y\_test, catboost\_y\_pred)

catboost\_conf\_matrix = confusion\_matrix(y\_test, catboost\_y\_pred)

catboost\_report = classification\_report(y\_test, catboost\_y\_pred, target\_names=['Not Churn', 'Churn'])

print("Best Parameters:", best\_catboost\_classifier)

print("Accuracy:", catboost\_accuracy)

print("F1-Score:", catboost\_f1)

print("Confusion Matrix:\n", catboost\_conf\_matrix)

print("Classification Report:\n", catboost\_report)

print("=" \* 80)

**10 CONCLUSION**

So, Finally I Conclude by saying that, this project “Telecom Churn Prediction Using Machine Learning” is very useful for Telecom industry and telecommunication plays a major role in day to day life. And it is very important for telecom industries to know the churners so they can improve services and can do better and grow in their sector.

**11 References**

**•** https://www.computer.org/csdl/proceedingsarticle/compsac/2012/4736a358/12OmNq G0SQD

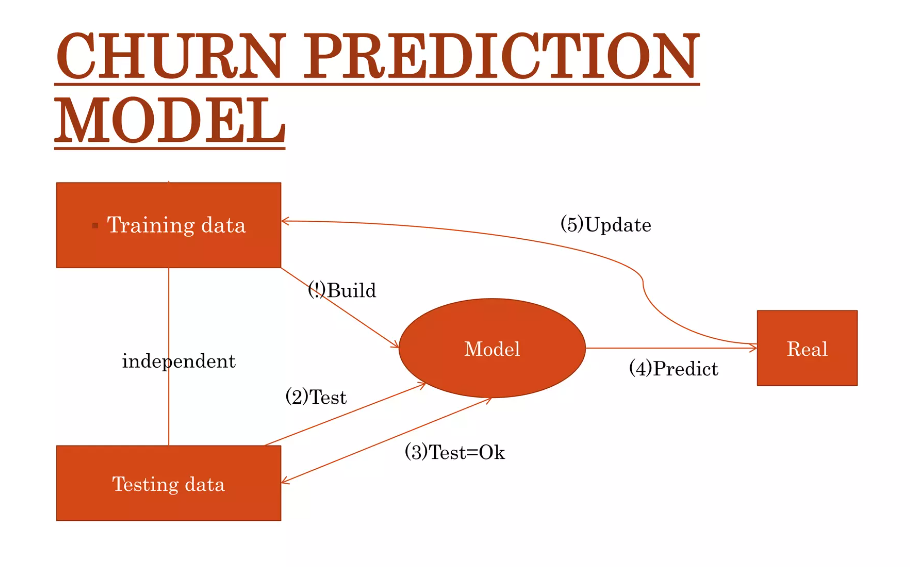
• [https://journalofbigdata.springeropen.com/articles/10.1186/s40537- 019-0191-6](https://journalofbigdata.springeropen.com/articles/10.1186/s40537-%20019-0191-6)

• https://towardsdatascience.com/telco-customer-churnrate-analysisd412f208cbbf

• https://www.computer.org/csdl/proceedingsarticle/hicss/2012/4525b023/12OmNApLGzX

• <https://www.geeksforgeeks.org/machine-learning/>

accurately.



Telecommunication plays a major role In day to day life. With increase in demand in the

telecom industry and there is though competition going among the telecom companies.

Customer churn is widely spread problem among the telecom industry. Customer churn has

huge impact on profits of the company. It will be great use for the companies to predict the

customer churn over a particular period of time. So that they do something that make the

customer not to change the network A study shows that there is increase in churn rate in the

telecom Companies. By using Machine Learning Algorithms We can perform predictions on

the data. In prediction of customer churn rate we are using Support Vector Machine(SVM)

and eXtreme Gradient Boosting (XGBoost) Algorithms.