CUSTOMER CHURN PREDICTION IN TELECOMMUNICATION SECTOR

A PROJECT REPORT

Submitted by

V.T.PRITHIBHA 731118205022

O.R.RAAGAMAALIKA 731118205023

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

GOVERNMENT COLLEGE OF ENGINEERING
ERODE-638 316.

ANNA UNIVERSITY: CHENNAI – 600 025

JUNE 2022

BONAFIDE CERTIFICATE

Certified that this Project Report "**CUSTOMER CHURN PREDICTION IN TELECOMMUNICATION SECTOR**" is the bonafide work of **V.T.PRITHIBHA (731118205022), O.R.RAAGAMAALIKA (731118205023)** who carried out the Project under my supervision.

SIGNATURE

Dr.P.KALYANI,M.E.,Ph.D.,
PROFESSOR,
HEAD OF THE DEPARTMENT,
DEPARTMENT OF IT,
GOVERNMENT COLLEGE OF ENGINEERING,
ERODE – 638 316.

SIGNATURE

Dr.R.POONGOTHAI,M.E.,Ph.D.,
SUPERVISOR,
ASSISTANT PROFESSOR,
DEPARTMENT OF IT,
GOVERNMENT COLLEGE OF ENGINEERING,
ERODE – 638 316.

Submitted for the University Examination held on _____at Government College of Engineering, Erode.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We wish to express our sincere gratitude to the persons who encouraged and helped us to complete our project successfully.

Our whole hearted thanks to **Dr.R.MURUGESAN., M.E., Ph.D.,** Principal, Government College of Engineering, Erode, for his constant encouragement and moral support during the course of the project.

We sincerely thank Prof. **Dr.P.KALYANI, M.E., Ph.D.,** Head of the Department, Department of Information Technology, Government College of Engineering, Erode, for furnishing every essential facility for doing our project and also for her valuable suggestion and Constant guidance throughout the project.

We sincerely thank our guide **Dr.M.POONGOTHAI, M.E., Ph.D.,** Assistant Professor, Department of Information Technology, Government College of Engineering, Erode, for his valuable help and guidance throughout the project.

We thank our class advisor **Dr.K.G.MAGESHWARI, M.TECH., Ph.D.,** Assistant Professor(Sr),Department of Information Technology, Government College of Engineering, Erode, and all other staff members of our department for their unending support and encouragement in completing our project.

ABSTRACT

Customer churn is defined as when customers stop doing business with a company or cancel his subscription. Customer churn prediction has gathered a greater interest in today's business, especially in telecommunications industries. Therefore, predicting customer churn in the telecommunication sector becomes an essential parameter for industry actors to protect their loyal customers. The primary objective is on the churn in telecom industries to accurately estimate the customer survival and customer hazard functions to gain the complete knowledge of churn over the customer tenure. Another objective is the identification of the customers who are at the blink of churn and approximating the time they will churn. Recently, there is a tremendous increase to apply ML techniques to predict customer churn in different industries. The model not only achieves a desirable prediction but also explains clearly about the selected features, and that a balanced relation between accuracy, recall, precision and explaining of customer churn prediction model.

AIM: To apply our core subject (BIG DATA, NEURAL NETWORKS) into a real worldusecase (TELECOMMUNICATION SECTOR).

KEYWORDS: Telecommunication Sector, Machine Learning, Churn Management, Prediction.

ARCHITECTURE OF PROPOSED SYSTEM

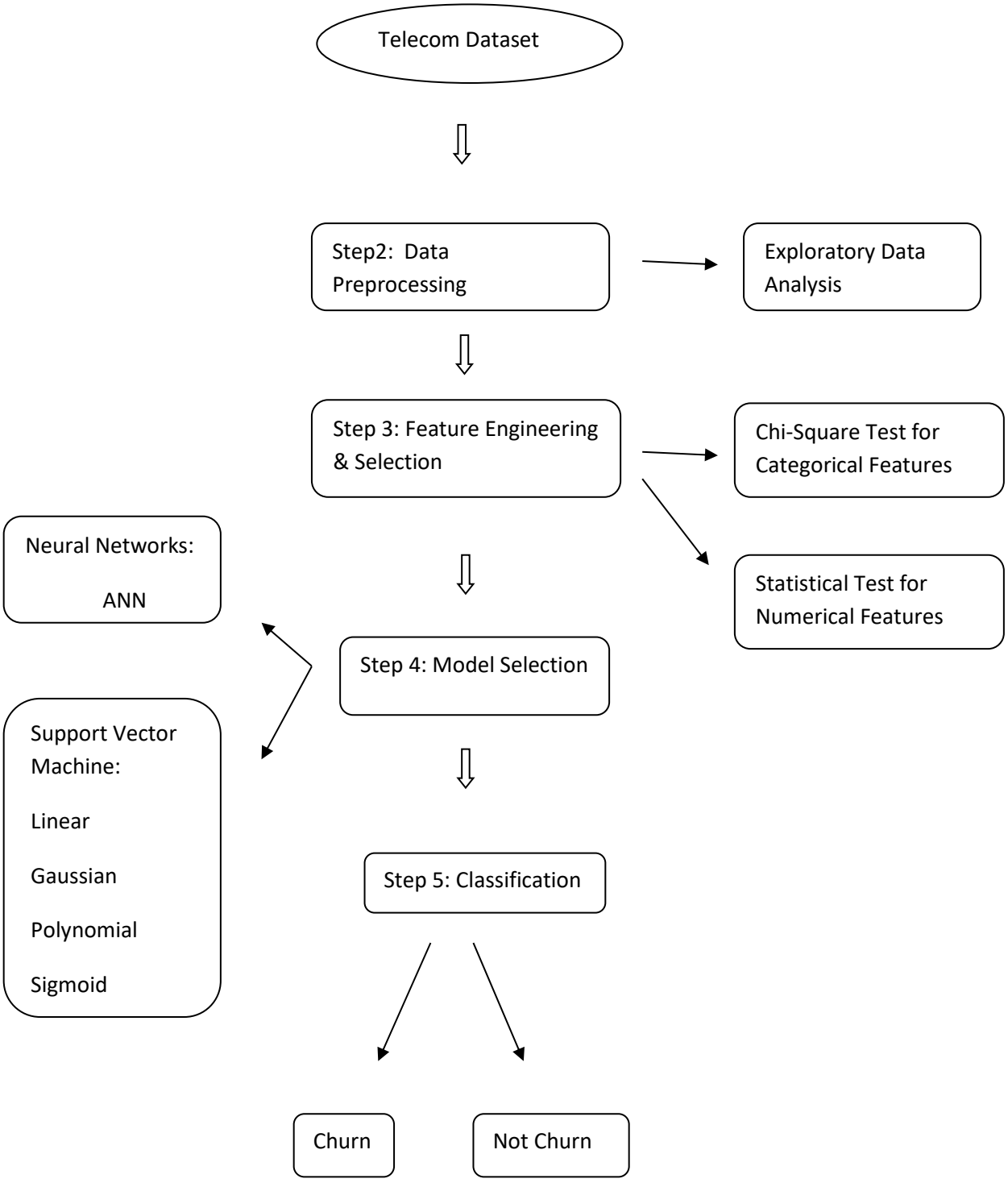

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

1.1INTRODUCTION

Churn is one of the most important service aspects in every industry. Churn is described as the activities of a customer's service being terminated, either by the customer or because of the customer's dissatisfaction with the service provided by a provider, or because of a more upgraded and affordable service provided by another supplier. The percentage of customers that discontinue using a company’s products or services during a particular time period is called a customer churn (attrition) rate. The customer churn is very complicated and the factors for churn, varies for each customer. Hence in this study, the customer churn is predicted by most recently proposed techniques that are analyzed. One of the ways to calculate a churn rate is to divide the number of customers lost during a given time interval by the number of acquired customers, and then multiply that number by 100 percent.

CHURN RATE = (Customers Lost/Total Customers) *100

1.2 CHURN PREDICTION

Churn prediction refers to identifying which customers are most likely to abandon a business or industry, or to cancel a service subscription. For many firms, it is a time-consuming process because recruiting new clients or consumers is generally more expensive than keeping old ones. Churn prediction can be thought of as a classification problem, with a client being classified as churning yes or no. Customers terminate their memberships for a variety of reasons, depending on their behavior and interests. As a result, it's vital to communicate with each of them on a regular basis if the company wants to keep them. Customer churn is a common

problem across businesses in many sectors. If you want to grow as a company, you have to invest in acquiring new clients.

Figure 1.2 Customer Churn

Every time a client departs, a large amount of money is lost. Knowing when a client is likely to depart and offering them incentives to stay can save a company a lot of money. Churn prediction is a necessary part of running a subscription business, and even minor swings in churn can have a big influence on the bottom line. "Is this customer going to quit us in X months?" Is it a yes or no question? It's a challenge of binary classification. The technique's accuracy is crucial to the success of any proactive retention campaign. Happy, engaged clients may be given special retention-focused offers or incentives, resulting in lower revenues for no reason. Churn forecasting model should be based on (almost) real-time data to quantify the risk of churning, not on static data. The objective is to build a model that predicts if a customer will churn or not using Python.

1.3 METHODOLOGIES

In order to find a possible solution to the problem of churn prediction i.e., successfully apply a machine learning technique to the available data, one needs a

deep understanding of the business rules of the telecommunications company and their specificity. The study indicates that machine learning techniques are mostly used and feature extraction is a very important task for developing an effective churn prediction model. Depending on the goal, researchers define what data they must collect. Next, selected data is prepared, preprocessed, and transformed in a form suitable for building machine learning models. Finding the right methods to training machines, fine-tuning the models, and selecting the best performers is another significant part of the work. Once a model that makes predictions with the highest accuracy is chosen, it can be put into production. The overall scope of work data scientists carries out to build ML-powered systems capable to forecast customer attrition may look like the following:

- Machine Learning Algorithms
  - Random Forest
  - Decision Tree
  - SVM
  - Logistic Regression
  - Gradient Boosting
- Neural Networks
- Deep Learning

Many different studies are conducted by researchers and telecom professional to construct churn prediction models with varying accuracy and precision on different data sets like Support Vector Machine (Linear and Kernel), Neural Networks. “As to identifying potential churners, machine learning algorithms can do a great job here. They reveal some shared behavior patterns of those customers who have

already left the company. Then, ML algorithms check the behavior of current customers against such patterns and signal if they discover potential churners."

Feature Learning

Feature learning is a set of approaches in machine learning that allows a system to learn the representations needed for feature identification or classification from raw data. Attempts to identify certain features algorithmically on real-world data such as photos, video, and sensor data have failed.

Optimization

Many learning issues are described as minimization of a loss function on a training set of instances, and machine learning is closely related to optimization. The difference between the model's predictions and the actual problem occurrences is expressed by the loss function.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE REQUIREMENTS

Processor	Intel Dual core
RAM	8GB
Hard disk capacity	5GB
Monitor	15-inch color

Table 2.1 Hardware Requirements

2.2 SOFTWARE REQUIREMENTS

Environment	JUPYTER NOTEBOOK
Coding Language	PYTHON
Operating System	WINDOWS 10
Data Set	KAGGLE DATA SET

Table 2. 2 Software Requirements

CHAPTER 3

SOFTWARE DESCRIPTION

3.1 PYTHON

Python is a dynamically semantic high-level interpreted programming language. Its high-level built-in data structures, together with dynamic typing and dynamic binding, are making it ideal for Rapid Application Development as well as usage as a scripting or glue language to link together existing components. Python's concise, easy-to-learn syntax encourages readability, lowering software maintenance costs. Modules and packages are available in Python. It boosts efficiency. It has a very rapid edit-test-debug cycle. A defect or incorrect input will never produce a segmentation fault, making debugging Python scripts simple. When the interpreter finds a mistake, it throws an exception. When a software fails to catch an exception, it indicates that there is no issue. The stack trace is printed by the interpreter. A virtual environment is a directory tree that contains Python executable files and other items that identify the environment as a digital one.

Advantages:

- ✓ Best for machine learning;
- ✓ cross-platform independence
- ✓ Python is Interactive and Interpreted, with Consistency and Simplicity.

3.2 JUPYTER NOTEBOOK

The original web application for producing and sharing computational documents is Jupyter Notebook. It provides a straightforward, simplified, and document-focused environment. It's an open-source web tool that lets data scientists create and share documents containing live code, equations, computational output,

visualizations, and explanatory text all in one place. Data cleansing and transformation, numerical simulation, exploratory data analysis, data visualization, statistical modeling, machine learning, deep learning, and other data science activities are all possible with them.

3.3LIBRARIES

3.3.1PANDAS

Pandas are an open-source data analysis and manipulation tool developed on top of the Python programming language that is quick, powerful, versatile, and simple to use. Pandas are a Python data analysis package. A strong and adaptable tool Pandas, a quantitative analysis tool, has become one of the most widely used Python packages. Pandas have become one of the most widely used Python libraries. Pandas are based on two key Python libraries: Matplotlib for data visualization and Numpy for arithmetic operations. Pandas acts as a wrapper around these libraries, enabling you to use fewer lines of code to access many of matplotlib's and NumPy's functions. Pandas'. Plot (), for example, integrates numerous Matplotlib methods into a single function, allowing you to plot a chart in only a few lines.

Features of Pandas:

- ✓ Excellent data representation
- ✓ Made for Python
- ✓ Extreme set of features
- ✓ Numerous built in functions
- ✓ Fast and efficient data frame

3.3.2 NUMPY

Numpy is becoming more popular, and it's now utilized in a variety of production systems for computing N-dimensional arrays. Numpy is a Python library with a lot of capability. In the industry, it's utilized for array computing. The fundamental features of the Numpy library will be discussed in this article. A multi-dimensional array and matrix data structures are included in Numpy. It can execute a variety of mathematical operations on arrays, including trigonometric, statistical, and algebraic algorithms. As a result, there are a lot of mathematical, algebraic, and transformation functions in the library. Numpy is a Numeric and num array extension. Numpy objects are widely used by Pandas objects. Pandas are essentially a Numpy extension.

Features of Numpy:

- ✓ High Performance
- ✓ Multidimensional Container
- ✓ Broadcasting Functions
- ✓ Work with varied database
- ✓ Additional linear algebra

3.3.3 MATPLOTLIB

In most cases, a Python Matplotlib script is constructed so that only a few lines of code are necessary to produce a visual data plot. The Matplotlib scripting layer combines two APIs: the pyplot API and the Matplotlib scripting layer. The pyplot API is a hierarchy of Python code objects topped by Matplotlib. Pyplot. A set of object-oriented API items that can be constructed with more freedom than pyplot. This API allows you to use Matplotlib backend layers directly. Compiling from source necessitates the presence of the necessary compiler for system OS, as well as

any dependencies, setup scripts, configuration files, and patches on the local system. As a result, the installation might be somewhat difficult.

Features of Matplotlib:

- ✓ Semantic way to generate complex, subplot grids
- ✓ Setting the aspect ratio of the axes box
- ✓ Colored labels in legends
- ✓ Ticks and labels
- ✓ 3D plots now support minor ticks

3.3.4 SEA BORN

Seaborn is a Python module for creating statistical visuals. It is based on Matplotlib and tightly interacts with pandas' data structures. Seaborn assists in exploring and comprehending given data. Its charting functions work with data frames and arrays that include whole datasets, doing the necessary semantic mapping and statistical aggregation internally to generate useful graphs. Its dataset-oriented, declarative API allows you to concentrate on the meaning of the charts rather than the mechanics of drawing them.

Features of Sea born:

- ✓ Built in themes for styling Matplotlib graphics
- ✓ Visualizing univariate and bivariate data
- ✓ Fitting in and visualizing linear regression models
- ✓ Plotting statistical time series data

3.3.5 TENSOR FLOW

Tensor Flow is an open-source machine learning platform that runs from start to finish. It contains a large, flexible ecosystem of tools, libraries, and community resources that researchers may use to solve problems. Tensor Flow enables creating machine learning models for desktop, mobile, web, and cloud easy for both beginners and professionals. With a rudimentary grasp of machine learning ideas and essential concepts, Tensor Flow is easy to employ.

Features of Tensor flow:

- ✓ Open-source Library
- ✓ Easy to run
- ✓ Fast Debugging
- ✓ Effective and Scalable

3.3.6 KERAS

Keras is a Deep Learning trend that is only getting started. It's a high-level neural network API that uses Tensor Flow and Theano as a foundation. Keras use its backend engines to handle the jobs that Keras is unable to do. It emphasizes quick experiments. The Model is its core ideology. Keras is the greatest Deep Learning library available. It's easy to read and understand. It's an easy-to-use Keras library.

Features of Keras:

- ✓ Modularity
- ✓ Large Dataset
- ✓ Evaluation and Prediction
- ✓ Train from Numpy Data and pre-trained models

CHAPTER 4

LITERATURE SURVEY

4.1 Overview

The majority of related research focused on using only one data mining method to extract knowledge, while some compared many strategies to predict churn. Several efforts in the telecom industry that use machine learning on a big data platform to anticipate client churn. Aside from that, this study looks into the methodologies used in machine learning algorithms for obtaining accurate forecast levels of telecom clients. The study's goal is to "deliver a solution for the prediction of customer churn in the telecom sector utilizing machine learning in a big data platform," according to the research.

4.2 Research Gap

For assessing churn prediction in the telecom business, Lemmens and Gupta (2013) used Stochastic Gradient Boosting models. In the telecom business, Kaur and Mahajan (2015) employed data mining and R techniques to forecast turnover rate. Gursoy (2010) revealed that using data mining technologies to anticipate customer attrition behavior in the telecom sector was successful. Data mining techniques such as decision trees and logistic regression were used by the author. In the telecom business, Adebiyi, et al (2016) employed logistic regression to investigate customer turnover and retention decisions. Poel and Lariviere (2004) employed neural networks, classification trees, and regression to forecast client attrition. Hadden et al. (2006) used Matlab, while Dwivedi et al. (2019) used SAS Enterprise Miner to estimate churn in the telecom industry.

CHAPTER 5

SYSTEM ANALYSIS

5.1 EXISTING SYSTEM

The model for churn prediction will be built using R programming in the existing system. It's a popular programming language used by statisticians and data miners to create statistical tools and do data analysis. R is a free and strong statistical analysis programme that has yet to be used to develop a model for churn prediction. There will be three main possibilities in the system, notably View the outcomes of applying logistic regression and decision trees on the supplied dataset in a performance analysis. Testing– to create a list of customers with a high likelihood of churning from the input data, providing that the input data's properties are the same as the given dataset used for training. Training and testing those results in the creation of a model as well as the generation of data If any other type of dataset is provided, it will be added to the churn list. Confusion matrix analysis is used to highlight the outcomes of employing logistic regression and decision trees on the provided dataset in performance analysis. The user can then submit data for testing the system, as long as the data attributes are the same as those used for training with the publicly available dataset.

Disadvantages:

- The prediction is done in R programming, which hasn't been used to develop a churn prediction model yet.
- In the current system, the system can only anticipate based on a few features.

5.1.1 Existing Telecom Churn

Author	Year	Algorithm
Das and Gondkar	2018	Artificial Neural Network
Gupta	2018	Hybrid Model ML Technique
Azeem and Usman	2018	ML SVM & Boosting
Ebrah and Elnasir	2019	Machine Learning Technique
Adebiyi	2016	Logistic Regression
Kaur and Mahajan	2015	Data Mining and R tool

Table 5.1.1Existing Telecom Churn

5.1.2Churn in Other Sectors

Data Set	Prediction Method
Subscriber Database	SVM, Random Forest, Logistic Regression
Pay-TV Company	Logistic Regression and Markov Chains random forest
Wireless telecom company	Decision Tree, Neural Network, K-means
Five mobile carriers	Logistic Regression
Taiwan Wireless Telecommunication Company	Decision Tree
Retailing Dataset	Neural Network, Logistic Regression

Table 5.1.2 Churn in Other Sectors

5.2 PROPOSED SYSTEM

MACHINE LEARNING

Machine learning is a discipline of artificial intelligence (AI) and computer science that focuses on using data and algorithms to teach machines how to learn in the same manner that human's do, with the goal of steadily improving accuracy. However, not all machine learning is statistical learning. A subset of machine learning is strongly related to computational statistics, which focuses on making predictions using computers.

5.2.1SUPPORT VECTOR MACHINE

The support vector machine (SVM), also known as support vector networks, is a collection of supervised learning algorithms for classification and regression models. Algorithm for SVM training. It's a non-probabilistic binary linear classifier since it creates a model that allocates new instances to one of two categories. Support-vector clustering (SVC) is a comparable methodology based on kernel functions that is suitable for unsupervised learning. The SVM algorithm's purpose is to find the optimum line or decision boundary for categorizing n-dimensional space into classes so that additional data points can be readily placed in the correct category in the future. A hyper plane is a boundary that represents the optimum decision.

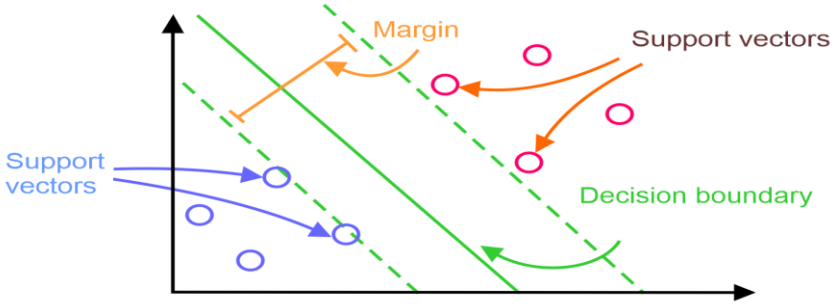

Figure 5.2.1Support Vector Machine

Classification

The difficulty of classification is determining which of a set of categories a new observation belongs to. To train an algorithm for classification problems, data scientists would use historical data with predetermined target variables, also known as labels (churn/non-churn) — answers that must be predicted. Businesses can use classification to address the following questions:

- Is this a churn-prone customer?
- Will a customer's subscription be renewed?
- Is it possible for a user to downgrade a pricing plan?

5.2.2 Neural Networks

Artificial neural networks or simulated neural networks are a subset of machine learning that are at the heart of deep learning methods. The human brain inspired their name and structure, which mimics how organic neurons communicate with one another. A neural network is an artificial intelligence technology that trains computers to analyse data in the same way that the human brain does. All decisions are made in the human brain via neural networks, which are made up of the basic building block Neuron and are naturally present in our bodies. The dendrites of a biological neuron are responsible for receiving data from other neurons, whereas the cell body sums all of the inputs and outputs the information through the cell body outside of the neuron is an axon. All communication and processing take place through synapses, which are connections between dendrites and the axon of the preceding neuron.

CHAPTER 6

DATA SET

6.1 EXPLORATORY DATA ANALYSIS

6.1.1 Data Collection

The Dataset is a fictional dataset of customer churn from telecom industry. Dataset source: https://www.kaggle.com/blastchar/telco-customer-churn. The features or attributes which define churn are as follows:

Customer ID: Customer ID, a unique ID for each customer

Gender: Whether the customer is a male or a female

Senior Citizen: Whether the customer is a senior citizen or not Partner: Whether the customer has a partner or not Dependents: Whether the customer has dependents or not Features based on the services opted by the Customer

Partner: Whether customer has a partner or not

Dependents: Whether customer has dependents or not

Tenure: Number of months the customer has used the services of the company / stayed with the company

Phone Service: Whether the customer has a phone service or not

Multiple Lines: Whether the customer has multiple phone lines or not

Internet Service: Type of Internet Connection opted by Customer. (DSL, Fiber optic, No)

Online Security: Whether the customer has online security or not

Device Protection: Whether the customer has device protection or not

Tech Support: Whether the customer has tech support or not

Streaming TV: Whether the customer has streaming TV or not

Streaming Movies: Whether the customer has streaming movies or not three numerical Features

Contract: The contract term of the customer (Month-to-month, One year, Two years)

Paperless Billing: Whether the customer has paperless billing or not

Payment Method: The customer's payment method (Electronic check, mailed check, Bank transfer (automatic), Credit card (automatic))

Monthly Charges: The amount charged to the customer monthly

Total Charges: The total amount charged to the customer

Churn: Whether the customer churn or not

6.1.2 Data Cleaning

The presence of noise, unknown or empty values, outliers, and erroneous values in the raw data may have a negative impact on the machine learning algorithm's performance. Data cleaning is used to reduce the number of inconsistencies in values, as well as noise and missing entries and characteristics. We determine the data types of the given attributes and the RAM used here because our dataset is suitably large. We discovered that there are no missing values or duplicate entries at this data cleaning stage.

6.1.3 Transformation of Data

It is discovered that when testing with various changes, data transformation strategies can greatly increase the overall performance of the churn prediction. Identifying the distinct values of each property and converting them to compute the churn rate, there is a need of a binary data type. There are empty numbers in total charges for a few tenure values. Find the number of missing vales and use the formula below to fill in those columns,

Tenure =Total Charges * Monthly Charges

6.1.4 Data Segmentation

Creating a train and test set from the dataset. The telecom dataset is divided into two subsets: one for training ML models and the other for evaluating the trained model's performance, known as the test set. The collection begins with 7043 instances, each representing a unique client, and 20 features, each representing an attribute. The split ratio is 75:25, which yields 5282 occurrences in the training set and 1760 instances in the test set.

6.2 FEATURE SELECTION OF DATASET

- Statistical Test
- Chi-Square Test

6.2.1 Target

The target value of this project is Churn.

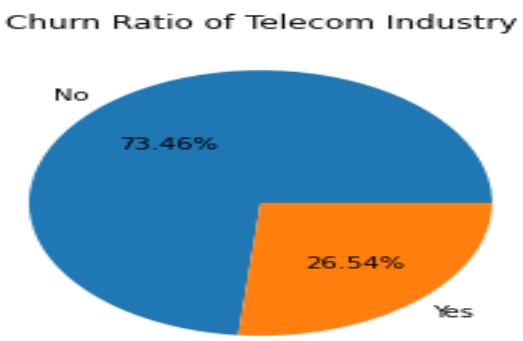

Figure 6.2. 1 Churn Ratio

26.54% of customers are churning, 73.46% of customers retain with our business.

6.2.2 STATISTICAL TEST

A statistical test is a tool that allows to make quantitative conclusions about a process or a group of activities. The main goal is to see if there is enough data to "reject" a theory or hypothesis about how the process works. This concept is known as the null hypothesis. The independent t-test is a statistical test that determines if two unrelated groups mean differ statistically significantly.

Numerical characteristics

A feature vector can be used to describe a numeric feature. A linear predictor function (connected to the perceptron) using a feature vector as input is one technique to achieve binary classification. Calculating the scalar product of the feature vector and a vector of weights qualifies those observations whose result above a certain threshold.

Result of the T-Test

The following are the outcomes of numerical features**:**

```
The T-Test result are :
Tenure is correlated with Churn | P-Value: 6.745011646431831e-146
MonthlyCharges is correlated with Churn | P-Value: 1.5695206211219092e-47
TotalCharges is correlated with Churn | P-Value: 5.629487289697648e-45


Features to be removed are : []
```

Figure 6.2.2 T-Test Results

6.2.3 Detection of Outliers

Outliers are observations that differ significantly (in terms of attributes) from the rest of the data points in a population sample. Graphing the characteristics or data points is the simplest technique to find an outlier. One of the finest and simplest ways to make inferences about the overall data and outliers is to use visualization.

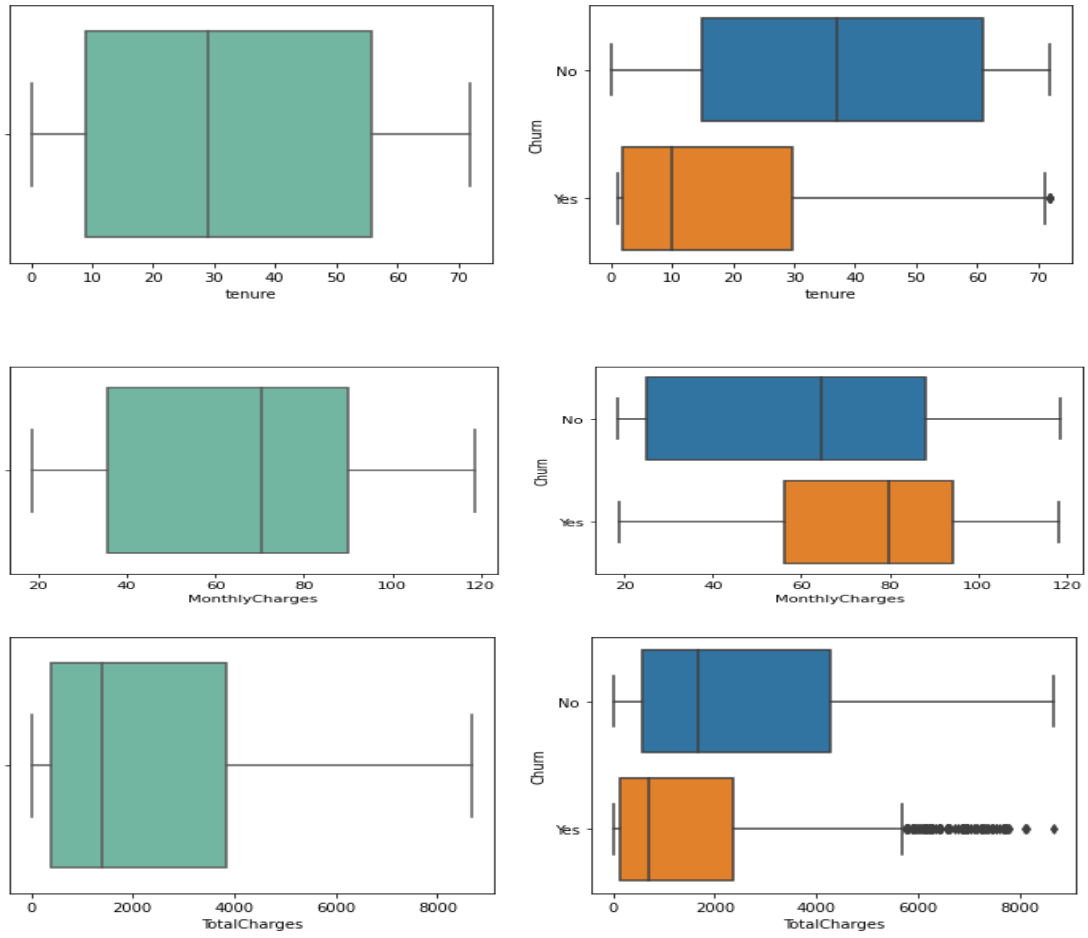

Figure 6.2. 3Outlier Detection of Numerical Features

RESULT OBSERVED:

- We can see from the left photos that there are no outliers in any of the three numerical features.
- We can see from the right images that Churn Customers have a very short tenure.
- Customer Churn Median The monthly fee is higher than customer who does not churn.
- Total Charges are lower for Churn Customers.

6.2.4Chi-Square Test

A chi-squared test (also chi-square or $\chi 2$ test) is a statistical hypothesis test that is valid to perform when the test statistic is chi-squared distributed under the null hypothesis, specifically Pearson's chi-squared test and variants thereof,

$$\boldsymbol{x^2 = \sum \frac{(Oi - Ei)^2}{Ei}}$$

x^2=Chi Squared, Oi=Observed Value, Ei=Expected value

P-Value

A p-value, or probability value, is a number which describes how likely it is that your data would have occurred under the null hypothesis of your statistical test**.** The p-value is a proportion: if the p-value is 0.05, that means that 5% of the time taken to see a test statistic at least as extreme as the one you found if the null hypothesis was true.

$$z = \frac{x - \mu}{\sigma}$$

σ= Standard Deviation, μ= Mean

Categorical features

Most machine learning algorithms only work with numeric values; many important real-world features are not numeric but rather categorical. As categorical features, they take on levels or values. The features are Customer ID, Gender, Senior Citizen, Partner, Dependents, Phone Service, Multiple Lines, Internet Service, Online Security, Online Backup, Device Protection, Tech Support, Streaming TV, Streaming Movies, Contract, Paperless Billing, and Payment Method.

Chi-Square test Result for Categorical Features

```
The ChiSquare result are :

customerID is  NOT  correlated with Churn | P-Value: 0.4935308220743649
Gender is  NOT  correlated with Churn | P-Value: 0.9133053640134722
SeniorCitizen is correlated with Churn | P-Value: 1.272738902968206e-26
Partner is correlated with Churn | P-Value: 3.2051580573988954e-26
Dependents is correlated with Churn | P-Value: 1.5736459837130873e-33
PhoneService is  NOT  correlated with Churn | P-Value: 0.11545819531090502
MultipleLines is correlated with Churn | P-Value: 0.004920729444037898
InternetService is correlated with Churn | P-Value: 7.816331813947432e-122
OnlineSecurity is correlated with Churn | P-Value: 3.1008969206083305e-142
OnlineBackup is correlated with Churn | P-Value: 1.7543681370437227e-100
DeviceProtection is correlated with Churn | P-Value: 1.0712098242548213e-91
TechSupport is correlated with Churn | P-Value: 7.966555837638945e-139
StreamingTV is correlated with Churn | P-Value: 2.9763744289949415e-62
StreamingMovies is correlated with Churn | P-Value: 1.5926291887016254e-62
ContractType is correlated with Churn | P-Value: 8.584733326965403e-193
PaperlessBilling is correlated with Churn | P-Value: 1.1159348460957409e-48
PaymentMethod is correlated with Churn | P-Value: 2.289591507114095e-106


Features to be removed are : ['customerID', 'Gender', 'PhoneService']
```

Figure 6.2. 4 Chi -Square Results

Observed Result

Customer ID, Gender, Phone Service features are to be removed as it does not correlate with churn's p-value. The remaining features are to be taken as it correlated with churn value and it is encoded. Correlation tests check whether variables are related without hypothesizing a cause-and-effect relationship.

Encoding

The other attributes have values that are of Yes or No types. The "Yes" values are encoded to binary value "1" and the "No" values are encoded to the binary value "0". The altered values are given below:

```
6661   0
4811   1
2193   0
Name: Churn, dtype: int64
```

6.2.5 Individual Attribute Classification

Taking each and every attribute individually and comparing those with churn to select the perfect feature which predicts churn optimally.

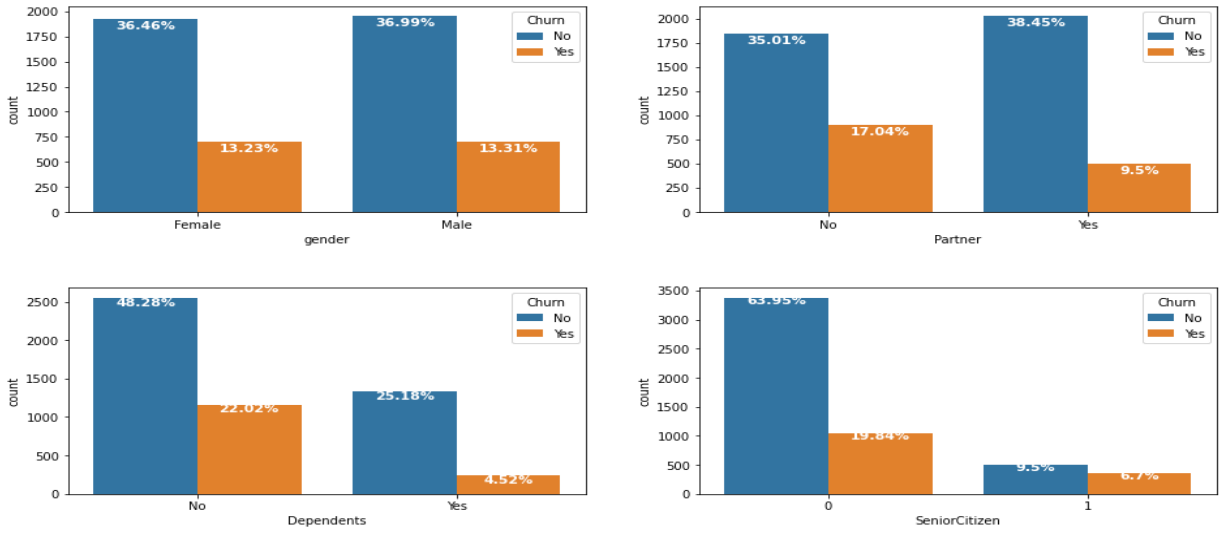

Figure 6.2.4. 1Gender partner Dependents Senior citizen

Observed Result

- The ratio of male and female customer is same
- Customers without Partner or dependent churn more than those who don't
- Senior Citizen's churn rate is low.

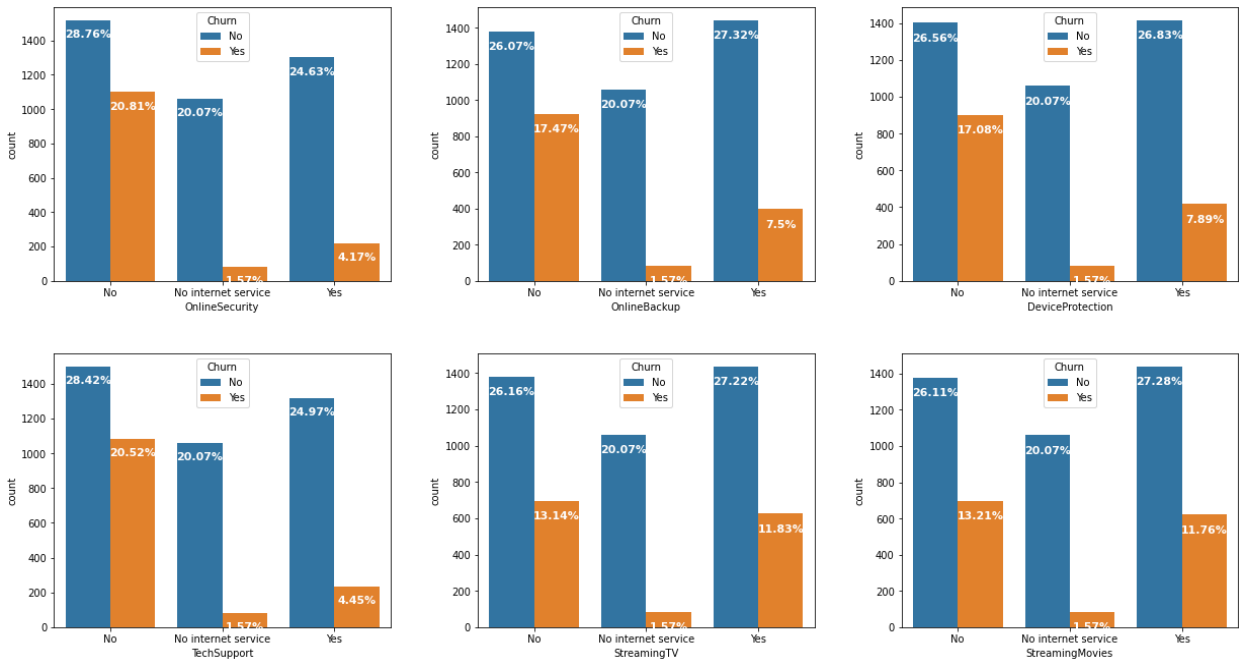

Figure 6.2.4. 2 Services Provided

Observed Result

- A Customer who has internet service but does not have any of the following is more likely to churn.
- "Online Backup"
- "Device Protection"
- "Online Security"
- "Tech Support"

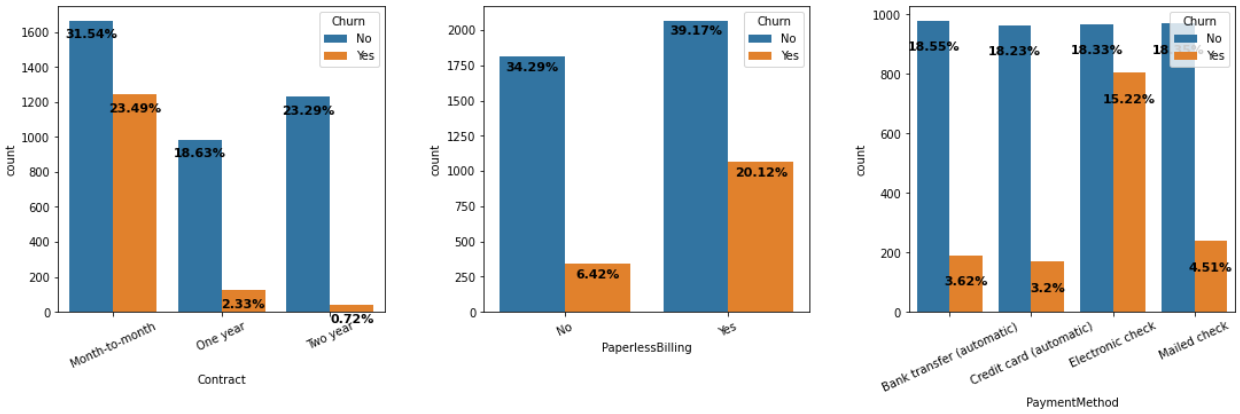

Figure 6.2.4. 3Contract Paperless billing Payment method

Observed Result

- Customer with "Fiber optic" have high churn rate.
- Customers with month-to-month contract have a high churn rate with 23.54%.
- Customer paying with electronic check also have a high churn rate around 15.35%.... this explain that why the churn of customers with paper billing is high.
- Check and cheque are examples of differences in spelling between British and American English.

CHAPTER 7

MODEL SELECTION

Comparative Analysis

- **Without Hyper Tuning**
  - ✓ Linear SVC
  - ✓ Gaussian RBF SVC
  - ✓ Polynomial SVC
  - ✓ Sigmoid SVC
- **With Hyper Tuning**
  - Linear SVC
  - Gaussian SVC
- **Artificial Neural Network**

7.1 LINEAR SVC

Linear SVC is used for linearly separable data, which means that if a dataset can be classified into two classes using a single straight line, it is considered linearly separable data, and the classifier employed is called Linear SVM. The training dataset provided with points of the type 1 or 1, with each identifying the class to which the point belongs. Any hyper plane can be expressed as a set of points fulfilling where is the normal vector to the hyper plane. The goal of a Linear SVC is to fit everything together returning a "best fit" hyper plane that divides or categorizes your data based on the data provided. In an effort to determine the predicted class after obtaining the hyper plane by feeding some features into it,

$$f(X) = w^T * X + b$$

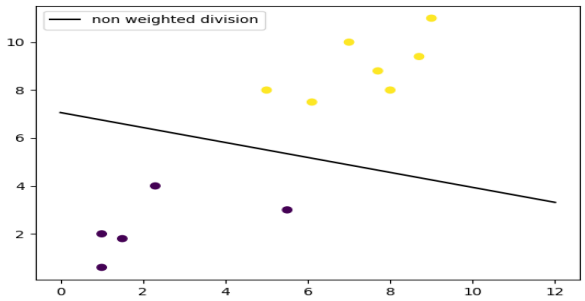

Figure 7. 1Linear SVC

7.2 GAUSSIAN RBF or NON-LINEARSVC

Due to its resemblance to the Gaussian distribution, Gaussian RBF kernels are the most generalized form of kernelization and one of the most extensively used kernels. For two points X1 and X2, the RBF kernel function computes their similarity, or how near they are to one other. This kernel can be expressed mathematically as follows,

f(X1, X2) = exp(-gamma * ||X1 - X2||^2)

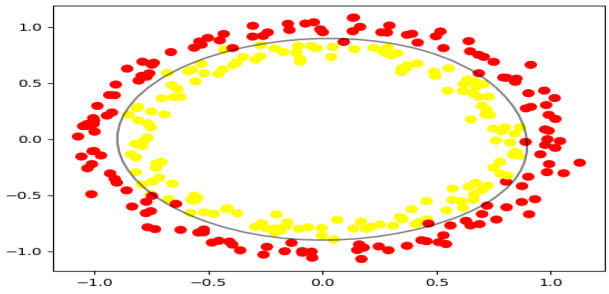

Figure 7. 2Gaussian RBF SVC

Where,

1. The variance is referred to as 'gamma,' and our hyper parameter is referred to as 'gamma.'

2. The Euclidean (L2-norm) Distance between two points is ||X1 - X2||. X1 and X2 are two different characters.

For non-linearly separated data, Gaussian RBF or Non-Linear SVM, also known as Kernel SVC, is used. This indicates that if a dataset cannot be classified using a straight line, it is classified as non-linear data, and the classifier employed is called Non-linear SVM classifier. The resulting technique is formally similar to Gaussian RBF kernels, except that every dot product is substituted by a nonlinear kernel function.

7.3 POLYNOMIAL SVC:

The polynomial kernel is a kernel function that allows learning of non-linear models and is often used with support vector machines and other kernelized models. It reflects the similarity of vectors in a feature space over polynomials of the original variables. Such combinations are referred to as interaction features in regression analysis. A polynomial kernel's (implicit) feature space is the same as that of polynomial regression, but without the combinatorial blowup in the number of parameters to learn. The features correspond to logical conjunctions of input features when the input features are binary-valued.

$$\mathbf{f\ (X1,\ X2) = (a + X1^{T} * X2)\ ^{\wedge}\ b}$$

7.4 SIGMOID SVC

The Sigmoid Kernel and the Multilayer Perceptron (MLP) kernel are other names for the Hyperbolic Tangent Kernel. The Sigmoid Kernel is derived from the

science of Neural Networks, where the bipolar sigmoid function is frequently utilized as an artificial neuron activation function.

$$f(X, y) = \tanh(\text{alpha} * X^T * y + C)$$

It's worth noting that a sigmoid kernel function SVM model is equivalent to a two-layer perceptron neural network. Because of its neural network theory origins, this kernel was highly popular for support vector machines. It has also been discovered that, while being just conditionally positive definite, it performs effectively in practice. The common value for alpha is 1/N, where N is the number of dimensions in the data.

7.5 ANN

Artificial neural networks, as shown in Fig.7.5, are a technology based on brain and nervous system research. These networks are modelled after biological neural networks, although they only employ a subset of the principles found in biological neural networks. ANN models, in particular, mimic the electrical activity of the brain and nervous system. Processing elements (also known as a neurode or a perceptron) are linked to one another. The neurode are usually organized in a layer or vector, with one layer's output acting as the input to the next layer and maybe further layers. A neurode can be linked to all or a subset of the neurode in the next layer, replicating brain synaptic connections. Data signals that have been weighted joining a neurode is a good way to get a feel for what it's like electrical stimulation of a nerve cell and, as a result, information flow within the network or brain A connection weight, w, n, m, is multiplied by the input values to a processing element, in, to replicate the strengthening of neural networks in the brain. In ANNs, learning is simulated by adjusting the strength or weights of the connections.

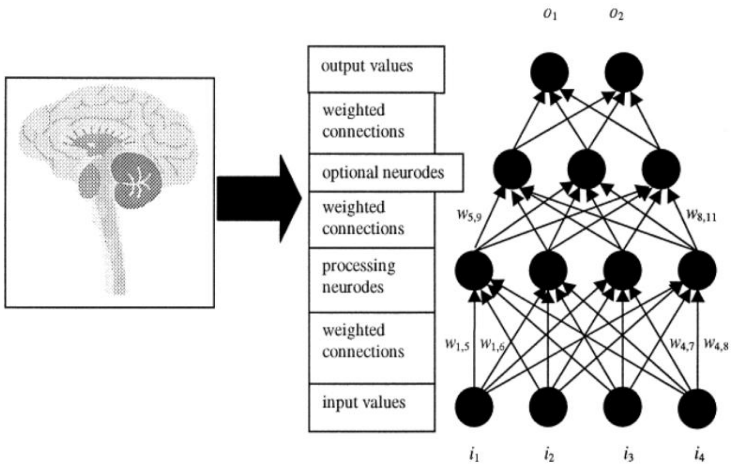

Figure 7. 5 ANN

7.6 HYPER PARAMETER TUNING

The term "hyper parameter tuning" refers to the process of selecting the best hyper parameters for a learning system. In an n-dimensional Euclidean space, a hyper plane is a flat, n-1-dimensional subset that separates the space into two separate portions. A hyper parameter is a model argument that is given a value before the learning process begins. It can be provided with improved hyper parameter values, increasing the predicted accuracy of your model. The model's over-fitting and under-fitting are controlled by hyper parameters. The best hyper parameters for different datasets are frequently different.

- Learning algorithms' generalization (test) error includes two primary components:
- Bias is an error that occurs as a result of simplifying model assumptions.
- Variance refers to the inaccuracy caused by the training set's randomness.

The trade-off between these elements is determined by the trade-off between these components is determined by the trade-off between these components is determined

the model's complexity and the amount of data used for training. Under-fitting (training and test error are both high) and over-fitting (training and test error are both high) can both be avoided with the right hyper parameters (Training error is low but test error is high).

7.6.1 TUNING PARAMETERS

C

The tolerance of misclassified observations is controlled by C. The lower C value provides for a wider margin at the tradeoff of lower accuracy and low performance variance. Higher C values, on the other hand, result in a narrower margin and better performance in categorizing most data points, indicating a large variance on the decision function.

GAMMA

It has an impact on how much the data affect the model's performance. The bigger the gamma value, the larger the radius, which takes into account more observations, whereas the smaller the gamma value, the smaller the radius. It was noted that a higher gamma would result in an over fitting issue.

No	Kernel function	Formula	Optimization parameter
1	Dot -product	$K(x_n, x_i) = (x_n, x_i)$	C
2	RBF	$K(x_n, x_i) = \exp(-\gamma \parallel x_n - x_i \parallel^2 + C$	C and γ

7.7 OPTIMIZER FOR ANN

The Adaptive Movement Estimation (Adam) Optimization technique is extended with the AdaMax algorithm. Is a generalisation of the Gradient Descent Optimization method. Adam's weights are updated in inverse proportion to the

scaled L2 norm (squared) of previous gradients. AdaMax takes this a step farther by include the so-called infinite norm (max) of prior gradients.Individual weights in Adam are updated by scaling their gradients inversely proportionate to a (scaled) L2 norm of their current and prior gradients.

The method has three hyperparameters, which are as follows:

• alpha: Initial step size (learning rate); 0.002 is a common setting.

• beta1: First momentum decay factor; a common value is 0.9.

• beta2: Infinity norm decay factor; a common value is 0.999.

The degeneration of beta1(t) decay schedule proposed in the study is computed using the original beta1 value raised to the power t, however different decay plans, such as keeping the value constant or declining more aggressively, might be utilised.

beta1(t) = beta1^t

CHAPTER 8

EVALUATION METRICS

The quality of a statistical or machine learning model is measured using evaluation metrics,

8.1 Accuracy

The ratio of the total number of correctly calculated predictions is called accuracy. To calculate classification accuracy, divide the total number of correct results by the total number of test data records and multiply by 100 to get the percentage.

$$\textbf{Accuracy} = (TP + TN) / (TP + FN + FP + TN)$$

8.2 Precision

The ratio of correctly classified cases to the total number of misclassified and properly classified cases is known as precision. The ratio of relevant cases to retrieved instances is known as precision, which is also known as positive predictive value.

$$\textbf{Precision} = TP / (TP + FP)$$

8.3 Recall

The proportion of correctly classified instances to the total number of correctly classified and unclassified cases is known as recall. The proportion of relevant examples retrieved among the total number of relevant instances is known as recall, which is also known as sensitivity.

$$\textbf{Recall} = TP / (TP + FN)$$

8.4 F1- Score

The F1-score combines the precision and recall measures, and is thought to be a good indicator of their relationship. As a result, this score accounts for both false positives and false negatives. It is not intuitively as simple to comprehend as precision.

F1 Score = 2 * (Precision ∗ recall) / (Precision + Recall)

8.5 Confusion Matrix

The confusion matrix is a matrix that is used to evaluate the classification models' performance for a given set of test data. Only if the true values for test data are known can it be determined. It's a specific table style that helps you to see how an algorithm, usually a supervised learning algorithm, performs. A confusion matrix is a summary of prediction results on a classification problem. It is an N *N matrix which is used for describing performance of classification model.

	Churn	**Not Churn**
Churn	True Positive	False Positive
Not Churn	False Negative	True Negative

Figure 8.5 Evaluation Matrix

False Positive: Customer is predicted as not churning but is actually churning.

False Negative: Customer was predicted to not churn, but it really churned**.**

True Positive: Customer Churn was predicted, but it was actually Customer Churn.

True Negative: Churn was predicted, but the customer was not really churned.

CHAPTER 9

IMPLEMENTATION

9.1 Source Code

```
import pandas as pd
importnumpyas np
importmatplotlib.pyplotasplt
%matplotlib inline
import seaborn as sns
import math
import warnings
warnings.filterwarnings("ignore")
fromsklearn.preprocessingimportMinMaxScaler
fromsklearn.model_selectionimporttrain_test_split
fromscipy.statsimportttest_ind
fromscipy.statsimport chi2_contingency
fromsklearn.model_selectionimportRandomizedSearchCV
fromsklearn.model_selectionimportcross_val_score
fromsklearn.metricsimportaccuracy_score, precision_score, recall_score, confusion_matrix
fromsklearn.metricsimportclassification_report, f1_score, fbeta_score, make_scorer
fromsklearn.svmimport SVC
df=pd.read_csv(r"C:\Users\raaga\Downloads\CHURN DATASET\WA_Fn-UseC_-Telco-Customer-Churn.csv")
df.head(5)
df.shape
X =df.drop(['Churn'], axis = 1)
Y =df['Churn']
```

```
x_train, x_test, y_train, y_test=train_test_split(X,Y, stratify = Y, test_size= 0.25, random_state= 42)

x_train.shape

train =pd.concat([x_train,  y_train], axis = 1)
train.shape

train =pd.concat([x_train,  y_train], axis = 1)
train.shape
```

Exploratory Data Analysis

```
train.info()

train_duplicates = train[train.duplicated()]

train_duplicates.shape

train.describe()

train.Churn.value_counts()

plt.figure(figsize=(6, 4))

plt.pie(train.Churn.value_counts()/train.shape[0], labels = train.Churn.unique(), autopct='%1.2f%%')

plt.title("Churn Ratio of Telecom Industry")

plt.show()
```

Viewing the Unique Values of Each Column

```
for feature in train.columns:

len_unique = len(train[feature].unique())

  if len_unique<= 10 :

uniq = train[feature].unique()
```

```
  else:
uniq = "too large to print all values.. some sample are" + str(train[feature][500:505].values)
print("Feature",feature,"has",len_unique,"values -",uniq)
TotalCharges_missing=train[pd.to_numeric(train['TotalCharges'], errors='coerce').isnull()]
TotalCharges_missing
train['total_services_opted']=(train[['PhoneService','MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']] == "Yes").sum(axis = 1)
train.loc[:,['Contract','tenure','MonthlyCharges','TotalCharges','total_services_opted']].sort_values(by=['tenure']).head(30)
print("Count of Missing Values in TotalChargesis",TotalCharges_missing.shape[0])
print("""",train.loc[488]['TotalCharges'],""")
TotalCharges_missing.index
for idx in TotalCharges_missing.index:
train.loc[idx,'TotalCharges'] = train.loc[idx]['tenure']*train.loc[idx]['MonthlyCharges']
print("Index: ", TotalCharges_missing.index[1],"- It's TotalCharges is",train.loc[TotalCharges_missing.index[1],'TotalCharges']
train['TotalCharges'] = train['TotalCharges'].astype(str).astype(float)
train['TotalCharges'].dtype
train.drop('total_services_opted', axis = 1, inplace = True)
```

```
categorical_features = []
numerical_features = []
for feature in train.columns:
  if train[feature].dtype == 'O':
categorical_features.append(feature)
  else:
numerical_features.append(feature)
numerical_features.remove("SeniorCitizen")
categorical_features.insert(4, "SeniorCitizen") # inseart 'SeniorCitizen' in 4th place
categorical_features
def count_percentage_subplots(features_list, rows, cols, huee, dataa, x_ticks_rotation = 0, figsize_row = 14, figsize_col = 9 , prcnt_color = 'white', prcnt_height = -100 ):
  fig = plt.figure(figsize = (figsize_row, figsize_col))
ax_list = []
  for i in range(1,cols * rows+1):
ax_list.append("ax"+str(i))
  for index,ax_name in enumerate(ax_list): # for features
ax_name = plt.subplot(rows, cols, index+1)
    feature = features_list[index]
sns.countplot(x=feature , hue = huee, data= dataa, order = sorted(list(dataa[feature].unique())))
plt.xticks(rotation= x_ticks_rotation)
```

```
    for index,p in enumerate(ax_name.patches):
      height = p.get_height()
      temp = list(round(dataa.groupby(huee)[feature].value_counts(sort = False)/len(dataa)*100,2))
ax_name.text(p.get_x()+p.get_width()/2., height+prcnt_height, str(temp[index]) + "%", horizontalalignment='center', fontsize=11, color=prcnt_color, weight = 'heavy')
 fig. tight_layout(pad=4.0)
plt.show()
count_percentage_subplots(['gender', 'Partner', 'Dependents', 'SeniorCitizen'],2,2, "Churn", train, figsize_row = 14, figsize_col =count_percentage_subplots(['PhoneService','MultipleLines'],1,2, "Churn", train, figsize_row = 12, figsize_col = 5 )
count_percentage_subplots(['InternetService'],1,1, "Churn", train, figsize_row = 7, figsize_col = 5 )
count_percentage_subplots(['OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies'],2,3, "Churn", train, figsize_row = 18, figsize_col = 10 )
count_percentage_subplots(['Contract','PaperlessBilling','PaymentMethod'],1,3, "Churn", train, figsize_row =16 ,figsize_col = 6 ,x_ticks_rotation = 25,prcnt_color = 'black')
for feature in numerical_features[0:3]:
  fig = plt.figure(figsize = (10,4))
  ax1 = plt.subplot(1,2,1)
```

```
  ax1.set_title(feature)
plt.hist(train[feature], bins =20)
 ax2 = plt.subplot(1,2,2)
plt.hist(train[train['Churn'] == 'Yes'][feature], bins = 20,alpha = 0.5, label = 'Churn', density = True)
plt.hist(train[train['Churn'] == 'No'][feature], bins = 20,alpha = 0.5, label = 'No Churn', density = True)
ax2.set_title(feature+"Distribution - Churn VS No-Churn")
 ax2.legend(loc = 'upper right')
plt.show()
 print("\n")
```

Feature Engineering

Renaming Some Feature

```
x_train.rename(columns={'gender':'Gender','Contract':'ContractType','tenure':'Tenure'},inplace=True)
x_test.rename(columns={'gender':'Gender','Contract':'ContractType'
,'tenure':'Tenure'  }
,inplace=True)
train.rename(columns={'gender':'Gender'
,'Contract':'ContractType','tenure':'Tenure'},inplace=True)
```

Feature Selection Using Statistical Test

```
def TTest(data, target, Continuous_features):
RemoveFeatures=[]
```

```
print('The T-Test result are : ')
   for feature in Continuous_features:
feature_data = data.groupby(target)[feature].apply(list)
pvalue = ttest_ind(*feature_data)[1]
       if (pvalue< 0.05):
print(feature, 'is correlated with', target , '| P-Value:', pvalue)
       else:
print(feature, 'is NOT correlated with', target , '| P-Value:', pvalue)
RemoveFeatures.append(feature)
   print("\n\n")
return(RemoveFeatures)
Continuous_features=['Tenure', 'MonthlyCharges', 'TotalCharges']
remove_features_con = TTest(data=train, target='Churn', Continuous_features = Continuous_features)
print("Features to be removed are :",remove_features_con)
```

CHI SQUARE TEST FOR CATEGORICAL FEATURE

```
categorical_features= list(x_train.columns)
for con in ['Tenure', 'MonthlyCharges', 'TotalCharges']:
categorical_features.remove(con)
def ChiSquare( data, target, categorical_features):
RemoveFeatures=[]
print('The ChiSquare result are : \n')
```

```
    for feature in categorical_features:
CrossTabResult=pd.crosstab(index = data[target], columns=data[feature])
pvalue = chi2_contingency(CrossTabResult)[1]
        if (pvalue< 0.05):
print(feature, 'is correlated with', target, '| P-Value:', pvalue)
        else:
print(feature, 'is \033[1m NOT \033[0m correlated with', target, '| P-Value:', pvalue)
RemoveFeatures.append(feature)
    print("\n\n")
return(RemoveFeatures)
remove_features_cat = ChiSquare(data = train, target = 'Churn', categorical_features
= categorical_features)
print("Features to be removed are :",remove_features_cat)
```

Dropping Variables

```
x_train.drop(remove_features_cat, axis = 1, inplace = True)
x_train.head(3)
x_test.drop(remove_features_cat, axis = 1, inplace = True )
```

Encoding of Categorical Features

```
y_train = y_train.map({'Yes':1,'No':0})
y_train.head(3)
y_test = y_test.map({'Yes':1,'No':0})
for feature in ['Partner', 'Dependents', 'PaperlessBilling']:
```

```
x_train[feature] = x_train[feature].map({'Yes':1,'No':0})

x_test[feature] = x_test[feature].map({'Yes':1,'No':0})

x_train['MultipleLines'] = x_train.MultipleLines.map({'Yes':1,'No':0,'No phone service':0})

x_train['ContractType'] = x_train.ContractType.map({'One year':1,'Two year':1,'Month-to-month':0})

x_test['MultipleLines'] = x_test.MultipleLines.map({'Yes':1,'No':0,'No phone service':0})

x_test['ContractType'] = x_test.ContractType.map({'One year':1,'Two year':1,'Month-to-month':0})

for feature in ['OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']:

x_train[feature] = x_train[feature].map({'Yes':1,'No':0,'No internet service':0})

x_test[feature] = x_test[feature].map({'Yes':1,'No':0,'No internet service':0})

x_train = pd.get_dummies(x_train, columns= ['InternetService','PaymentMethod'], drop_first = True)

x_test = pd.get_dummies(x_test, columns= ['InternetService','PaymentMethod'], drop_first = True)
```

Scaling of Numerical Features

```
TotalCharges_missing_x_train = x_train[pd.to_numeric(x_train['TotalCharges'], errors='coerce').isnull()]

for idx in TotalCharges_missing_x_train.index:

x_train.loc[idx,'TotalCharges']=x_train.loc[idx]['Tenure']*
x_train.loc[idx]['MonthlyCharges']
```

```
x_train['TotalCharges'] = x_train['TotalCharges'].astype(str).astype(float)

x_train['TotalCharges'].dtype

TotalCharges_missing_x_test = x_test[pd.to_numeric(x_test['TotalCharges'], errors='coerce').isnull()]

for idx in TotalCharges_missing_x_test.index:

x_test.loc[idx,'TotalCharges']=x_test.loc[idx]['Tenure']* x_test.loc[idx]['MonthlyCharges']

x_test['TotalCharges'] = x_test['TotalCharges'].astype(str).astype(float)

x_test['TotalCharges'].dtype

scaler = MinMaxScaler()

minmax_scaler = scaler.fit(x_train[['Tenure','MonthlyCharges','TotalCharges']])

x_train[['Tenure','MonthlyCharges','TotalCharges']] = minmax_scaler.transform(x_train[['Tenure','MonthlyCharges','TotalCharges']])

x_train.head(3)

x_test[['Tenure','MonthlyCharges','TotalCharges']] = minmax_scaler.transform(x_test[['Tenure','MonthlyCharges','TotalCharges']])
```

Model Selection

```
def plot_confusion_matrix(y_test, pred):
  cm = confusion_matrix(y_test, pred)

df_cm = pd.DataFrame(cm, [0,1], [0,1])

plt.figure(figsize=(5,3))

sns.set(font_scale=1.4) # for label size
```

```
sns.heatmap(df_cm, annot=True, fmt='g', cbar=False, cmap="YlGnBu") # font size
, annot_kws={"size": 16}
plt.title('Confusion Matrix\n', y=1.1)
plt.ylabel('Actual label\n')
plt.xlabel('Predicted label\n')
plt.show()
df_cm = pd.DataFrame([[1000,200],[150,250]],[0,1], [0,1])
annot_arr = np.array([['True Negative','FalsePostive'],['False Negative','TruePostive']], dtype = str)
plt.figure(figsize=(5,3))
sns.set(font_scale=1.4) # for label size
sns.heatmap(df_cm, annot=annot_arr, fmt='s', cbar=False, cmap="YlGnBu") # font size , annot_kws={"size": 16}
plt.title('Confusion Matrix\n', y=1.1)
plt.ylabel('Actual label\n')
plt.xlabel('Predicted label\n')
plt.show()
```

Metrics for Model Evaluation:

```
def model_metrics(table_name,model_name, y_test, pred, print_cm , print_cr):
  if print_cm == True:
plot_confusion_matrix(y_test, pred)
    print("\n")
  if print_cr == True:
```

```
print(classification_report(y_test, pred),"\n")
  accuracy = round( accuracy_score(y_test, pred),4)
  precision = round(precision_score(y_test, pred),4)
  recall = round(recall_score(y_test, pred),4)
  f1 = round(f1_score(y_test, pred),4)
table_name.loc[table_name.shape[0]] = [model_name ,accuracy, precision, recall, f1]
model_performance_without_hypertuning = pd.DataFrame(columns = ['model_name' ,'accuracy', 'precision', 'recall', 'f1'])
model_performance_after_hypertuning = pd.DataFrame(columns = ['model_name' ,'accuracy', 'precision', 'recall', 'f1'])
models_without_hp = []
```

#linear separation kernal (Linear kernal)

```
models_without_hp.append(('Linear SVC', SVC(kernel = 'linear', random_state = 42)))
```

#non-linear separation kernals(Gaussian RBF, Sigmoid, Polynomial)

```
models_without_hp.append(('Gaussian-rbf SVC', SVC(kernel = 'rbf', random_state = 42)))
models_without_hp.append(('polynomial SVC', SVC(kernel = 'poly', random_state = 42)))
models_without_hp.append(('Sigmoid SVC', SVC(kernel = 'sigmoid', random_state = 42)))
for name, model in models_without_hp:
```

```
  md = model.fit(x_train, y_train)
  pred = md.predict(x_test)
model_metrics(model_performance_without_hypertuning, name, y_test, pred, False, False)
model_performance_without_hypertuning
```

Hyper Paramaeter Tuning of models:

Linear SVC

```
import time
st = time.time()
linear_SVC = SVC(kernel = 'linear', random_state = 42)
param_dist = {    'C' : [0.001, 0.01, 0.1, 1.0, 10.0, 100.0]}
linear_SVC_hp= RandomizedSearchCV(linear_SVC, param_distributions = param_dist  , cv = 5, random_state= 42)
result2 = linear_SVC_hp.fit(x_train, y_train)
print(result2.best_estimator_)
pred_2 = result2.predict(x_test)
model_metrics(model_performance_after_hypertuning, 'Linear SVC', y_test, pred_2, True, True)
et = time.time()
elapsed_time = et - st
print('Execution time:', elapsed_time, 'seconds')
print(model_performance_after_hypertuning.tail(1))
```

Gaussian-rbf SVC

```
import time
```

```
st = time.time()
Gaussian_rbf_SVC = SVC(kernel = 'rbf', random_state = 42)
param_dist = {    'C' : [0.001, 0.01, 0.1, 1.0, 10.0, 100.0],    'gamma' : [ 0.0001 , 0.001, 0.01, 0.1, 1.0, 10]}
Gaussian_rbf_SVC_hp=RandomizedSearchCV(Gaussian_rbf_SVC,
param_distributions = param_dist  , cv = 5, random_state= 42)
result3 = Gaussian_rbf_SVC_hp.fit(x_train, y_train)
et = time.time()
elapsed_time = et - st
print('Execution time:', elapsed_time, 'seconds')
print(result3.best_estimator_)
pred_3 = result3.predict(x_test)
model_metrics(model_performance_after_hypertuning, 'Gaussian-rbf SVC', y_test, pred_3, True, True)
print(model_performance_after_hypertuning.tail(1))
```

MODEL BUILDING USING NEURAL NETWORKS

```
from sklearn.metrics import confusion_matrix,classification_report
from sklearn.metrics import accuracy_score,recall_score,precision_score,f1_score
from keras.layers import BatchNormalization
from keras.models import Sequential
from keras.layers import Dropout
from keras.layers import Dense
import matplotlib.pyplot as plt
```

```
import seaborn as sns
import pandas as pd
import numpy as np
import collections
import warnings
warnings.filterwarnings("ignore")
pip install numpy --upgrade
df.dtypes
df["TotalCharges"] = pd.to_numeric(df["TotalCharges"],errors="coerce")
df.isnull().sum()
summary = df.describe(include=["O"])
summary
df.dropna(how="any",inplace=True)
df
df.isnull().sum()
data_cont = ['tenure','MonthlyCharges', 'TotalCharges']
data_cat = [ 'SeniorCitizen', 'Partner', 'Dependents','MultipleLines', 'InternetService','OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport','StreamingTV','StreamingMovies','Contract', 'PaperlessBilling','PaymentMethod']
scaler = StandardScaler()
data_continuous = scaler.fit_transform(df[data_cont])
for cols in data_cat:
```

```
df.loc[:,cols] = LabelEncoder().fit_transform(df.loc[:,cols])

onehotencoder = OneHotEncoder(sparse=False)

data_categorical = onehotencoder.fit_transform(df[data_cat])

features = np.concatenate([data_continuous, data_categorical], axis=1)

target = df.iloc[:,20:].values

target = LabelEncoder().fit_transform(target)

X_train, X_test, y_train, y_test = train_test_split(features, target, test_size = 0.2, random_state = 0)

import time

st = time.time()
classifier = Sequential()

classifier.add(Dense(units=6,kernel_initializer="uniform",activation="relu",input_shape=(46,)))

classifier.add(BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001, center=True, scale=True))

classifier.add(Dense(units=6,kernel_initializer="uniform",activation="relu"))

classifier.add(BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001, center=True, scale=True))

classifier.add(Dense(units=6,kernel_initializer="uniform",activation="relu"))

classifier.add(BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001, center=True, scale=True))

classifier.add(Dropout(0.2))

classifier.add(Dense(units=1,kernel_initializer="uniform",activation="sigmoid"))
```

```
classifier.compile(optimizer = 'adamax', loss = 'binary_crossentropy', metrics = ['accuracy'])
et = time.time()
elapsed_time = et - st
print('Execution time:', elapsed_time, 'seconds')
classifier.fit(X_train, y_train, batch_size = 50, epochs = 150)
classifier.evaluate(X_test, y_test)
y_pred = classifier.predict(X_test)
y_pred = (y_pred> 0.5)
print(classification_report(y_test,y_pred))
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm.T,square=True,annot=True,fmt='d',
cbar=False,xticklabels=['No','Yes'],yticklabels=['No','Yes'] )
plt.xlabel('true label')
plt.ylabel('predicted label')
print('Accuracy: %.4f' % accuracy_score(y_test, y_pred))
print('Precision: %.4f' % precision_score(y_test, y_pred))
print('Recall: %.4f' % recall_score(y_test, y_pred))
print('F1-Score: %.4f' % f1_score(y_test, y_pred))
print('Execution time:', elapsed_time, 'seconds')
```

9.2 Output Screenshots

Models with hyper tuning

	model_name	accuracy	precision	recall	f1
0	Linear SVC	0.7922	0.6278	0.5310	0.5754
1	Gaussian-rbf SVC	0.7939	0.6421	0.5032	0.5642
2	polynomial SVC	0.7774	0.6100	0.4454	0.5149
3	Sigmoid SVC	0.7087	0.4533	0.4775	0.4651

Models without hyper tuning

	model_name	accuracy	precision	recall	f1
0	Linear SVC	0.7922	0.6278	0.5310	0.5754
1	Gaussian-rbf SVC	0.7944	0.6496	0.4882	0.5575

Artificial Neural Network Model

```
Accuracy: 0.8024
Precision: 0.6655
Recall: 0.4959
F1-Score: 0.5683
Execution time: 3.2178173065185547 seconds
```

CHAPTER 10

CONCLUSION AND FUTURESCOPE

10.1 CONCLUSION

Customer churn is a crucial activity in rapidly growing and competitive telecommunication sector, due to the high cost of acquiring new customers. Churn prediction has emerged as indispensable part of strategic decision making and planning process. In this study also investigated the performances of four different algorithms namely Linear SVC Gaussian-rbf SVC, polynomial SVC and Sigmoid SVC and along with that we also tried Artificial Neural Networks with different Optimizer (Adamax) and compared performance of all four Evaluation metrics namely Accuracy, Precision, Recall and F1-score and additionally we calculated Execution time for all algorithms and came under conclusion that model using ANN found to be an optimal model when compared with other models whose accuracy and execution time is found to be high.

10.2 FUTURE SCOPE

Hence our future scope is that we decided to go for Hybrid model of combining two different architectures namely Support Vector Machine and Neural Networks which is known to Support Vector Neural Network (SVNN).

CHAPTER 11

REFERENCES

1] Churi, A., Divekar, M. and Mahe, R., 2015. Prediction Of Customer Churn In Mobile Industry Using Probabilistic Classifiers. International Journal of Advance Foundation And Research In Science & Engineering, 1(10), pp.41-49.

2] Khan, Y., Shafiq, S., Naeem, A., Ahmed, S., Safwan, N. and Hussain, S., 2019. Customers churn prediction using artificial neural networks (ANN) in telecom industry. International Journal of Advanced Computer Science and Applications, 10(9).

3] Khan, Y., Shafiq, S., Naeem, A., Ahmed, S., Safwan, N. and Hussain, S., 2019. Customers churn prediction using artificial neural networks (ANN) in telecom industry. International Journal of Advanced Computer Science and Applications, 10(9).

4] Yıldız, M. and Albayrak, S., 2017. Customer churn prediction in telecommunication with rotation forest method. DBKDA 2017, p.35.

5] Vafeiadis, T., Diamantaras, K.I., Sarigiannidis, G. and Chatzisavvas, K.C., 2015. A comparison of machine learning techniques for customer churn prediction. Simulation Modelling Practice and Theory, 55, pp.1-9.

6] Dahiya, K. and Talwar, K., 2015. Customer churn prediction in telecommunication industries using data mining techniques-a review. Int. J. Adv. Res. Comput. Sci. Softw. Eng, 5(4), pp.417-433.

7] Saini, N., Monika, G.K. and Garg, K., 2017. Churn prediction in telecommunication industry using decision tree. Int J Eng Res Technol, 6.