

PROJECT REPORT - IBM BUILD-A-THON
Telecom Customer Churn Prediction using Watson Auto AI

presented by

Vignesh K

email-id: vignesh.20cs@licet.ac.in

in the month of

October 2020

TABLE OF CONTENTS

1	INTRODUCTION
1.1	Overview
1.2	Purpose
2	LITERATURE SURVEY
2.1	Existing Problem
2.2	Proposed Solution
3	THEORITICAL ANALYSIS
3.1	Block Diagram
3.2	Hardware / Software Designing
4	EXPERIMENTAL INVESTIGATIONS
5	FLOWCHART
6	RESULT
7	ADVANTAGES & DISADVANTAGES
8	APPLICATIONS
9	CONCLUSION
10	FUTURE SCOPE
11	BIBILIOGRAPHY
	APPENDIX
A.	Source Code

INTRODUCTION

1.1 OVERVIEW

Churn prediction is one of the most popular Big Data use cases in business. It consists of detecting customers who are likely to cancel a subscription to a service. This can be telecom companies, SaaS companies, and any other company that sells a service for a monthly fee.

In the telecom industry, customers can choose from multiple service providers and actively switch from one operator to another. In this highly competitive market, the telecommunications industry experiences an average of 15-25% annual churn rate. Given the fact that it costs 5-10 times more to acquire a new customer than to retain an existing one, customer retention has now become even more important than customer acquisition. For many incumbent operators, retaining high profitable customers is the number one business goal.

1.2 PURPOSE

Customer churn prediction can help you see which customers are about to leave your service so you can develop proper strategy to re-engage them before it is too late. This is a vital tool in a business' arsenal when it comes to customer retention. Having the ability to accurately predict future churn rates is essential because it helps your business gain a better understanding of future expected revenue. Predicting churn rates can also help your business identify and improve upon areas where customer service is lacking.

To reduce customer churn, telecom companies need to predict which customers are at high risk of churn.

In this project, the customer-level data of a leading telecom firm, build predictive models to identify customers who will stay in the company (or) who will leave the company based on a set of parameters.

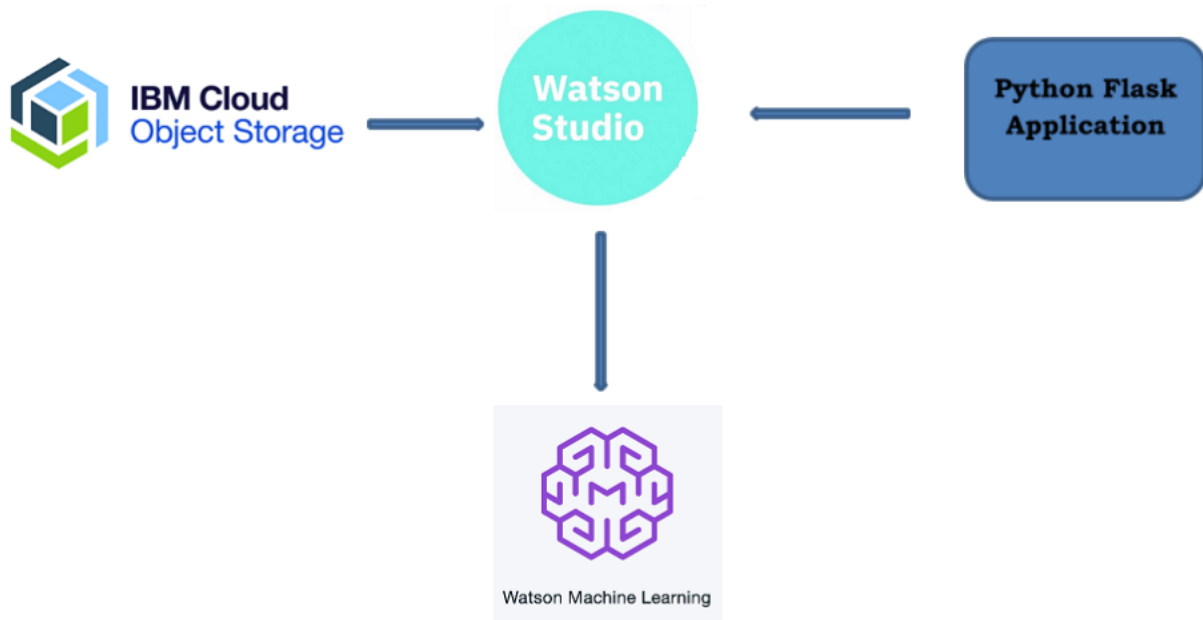
LITERATURE SURVEY

S.No	Title & Author	Existing Problem	Proposed Solution
1.	Telco Churn Prediction with Big Data - Yiqing Huang, Fangzhou Zhu, Mingxuan Yuan, Ke Deng, Yanhua Li, Bing Ni, Wenyuan Dai, Qiang Yang, Jia Zeng	A churning customer quits the service provided by operators and yields no profit any longer. The prepaid customers have a significantly higher churn rate (on average 9.4%) than the postpaid customers (on average 5.2%), because the prepaid customers are not bound to contract and can quit easily without recharging. The cost of acquiring new customers is much higher than that of retaining the existing ones	Experimental results confirm that the prediction performance has been significantly improved by using a large volume of training data, a large variety of features from both business support systems (BSS) and operations support systems (OSS), and a high velocity of processing new coming data. Automatic matching retention campaigns with the targeted potential churners significantly boost their recharge rates, leading to a big business value.
2.	On the Operational Efficiency of Different Feature Types for Telco Churn Prediction - Sandra Mitrović, Bart Baesens, Wilfried Lemahieu, Jochen De Weerd	Existing studies on customer churn prediction generally agree that adding features typically increases predictive performance, they rarely discuss the accompanying issues such as data availability and computational cost.	The gap between predictive performance and operational efficiency by devising a new feature type classification and a novel reusable method to determine optimal feature type combinations based on Pareto multi-criteria optimization. The results provide several insights that can serve as a guideline for industry practitioners.
3.	A Proposed Churn Prediction Model - Essam Shaaban, Yehia Helmy, Ayman Khedr, Mona Nasr	Conventional churn prediction techniques have the advantage of being simple and robust with respect to defects in the input data, they possess	The purpose of prediction is to anticipate the value that a random variable will assume in the future or to estimate the likelihood of future events.

		serious limitations to the interpretation of reasons for churn. Therefore, measuring the effectiveness of a prediction model depends also on how well the results can be interpreted for inferring the possible reasons of churn.	
4.	Handling Imbalanced Data in Customer Churn Prediction Using Combined Sampling and Weighted Random Forest - Veronikha Effendy, Adiwijaya, Z.K.A. Baizal.	At the time of the customer churn is taking place, the percentage of data that describes the customer churn is usually low. Unfortunately, the churn data is the data which have to be predicted earlier. The lack of data on customer churn led to the problem of imbalanced data. The imbalanced data caused difficulties in developing a good prediction model.	This research applied a combination of sampling techniques and Weighted Random Forest (WRF) to improve the customer churn prediction model on a sample dataset from a telecommunication industry in Indonesia. Sampling techniques were applied to enhance performance.
5.	Churn Prediction using Dynamic RFM-Augmented node2vec - Sandra Mitrovic, Bart Baesens, Wilfried Lemahieu, Jochen De Weerd.	The suggested approaches demonstrate a lot of creativity when it comes to deriving new features from the underlying networks, they also exhibit at least one of the following problems: they either do not account properly for dynamic aspects of call networks or they do not exploit the full potential of joint interaction and structural features and additionally, they usually address these in a non-systematic manner which involves hand-engineering of features.	<ol style="list-style-type: none"> 1. Slicing a monthly call graph to capture dynamic changes in calling patterns. 2. A devise network designs which conjoin interaction and structural information. 3. Adapted and applied the node2vec method to learn node representations in a more automated way and to avoid the need for feature handcrafting.

THEORITICAL ANALYSIS

3.1 BLOCK DIAGRAM



The block diagram depicts the workflow of the entire system. Watson Studio acts the central point of computation, and is used for running python notebooks and creating, monitoring, and managing deployments. The runtime environment is powered by Watson Machine Learning Service. The UI is designed using HTML and the backend process is automated using Flask framework, which also facilitates deployment of the ML models using the scoring endpoint.

3.2 HARDWARE / SOFTWARE DESIGNING

The following are the hardware requirements for standard users (commodity hardware) of the proposed system:

Processor: Core i5 Quad Core

RAM:8GB

The software specification for the proposed system is as follows:

IBM Watson Studio:

Watson ML Package - 'Lite'

Instance Type - 'v2'

Environment Definition - Default Python 3.6XS

Virtual Hardware Configuration - 2 vCPU 8GB RAM

COS Instance Region - 'London'

Python Flask Application:

HTML - 5.0

Flask - 1.1.2

Python Libraries required:

scikit-learn

pandas

numpy

seaborn

json

sklearn.preprocessing

sklearn.model_selection

sklearn.feature_selection

EXPERIMENTAL INVESTIGATIONS

3.1 LOADING THE DATASET

The dataset is provided in the template provided. This data set contains details of a company's customers and the target variable is a binary variable reflecting the fact whether the customer left the company (or) he continues to be a customer.

IBM has provided options to add the dataset as an asset into the projects. The project assets can be added directly into the Python notebook using a simple process, and the code is automatically generated.

The dataset is available [here](#).

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0
5	6	15574012	Chu	645	Spain	Male	44	8	113755.78	2	1	0	149756.71	1
6	7	15592531	Bartlett	822	France	Male	50	7	0.00	2	1	1	10062.80	0
7	8	15656148	Oblinna	376	Germany	Female	29	4	115046.74	4	1	0	119346.88	1
8	9	15792365	He	501	France	Male	44	4	142051.07	2	0	1	74940.50	0
9	10	15592389	H?	684	France	Male	27	2	134603.88	1	1	1	71725.73	

The metadata of the dataset is as follows:

- # of rows: 10,000
- # of columns: 14
- # of Input Variables: 13
- # of Output Variable(s): 1 - ["Exited"]

3.2 INFORMATION ABOUT DATASET

The info() function gives some of the basic details about the dataset. It gives the information about the following:

- Number of entries
- Null Value status
- Datatype

for each attribute of the dataframe.


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
RowNumber          10000 non-null int64
CustomerId          10000 non-null int64
Surname            10000 non-null object
CreditScore        10000 non-null int64
Geography          10000 non-null object
Gender             10000 non-null object
Age               10000 non-null int64
Tenure            10000 non-null int64
Balance           10000 non-null float64
NumOfProducts     10000 non-null int64
HasCrCard          10000 non-null int64
IsActiveMember    10000 non-null int64
EstimatedSalary   10000 non-null float64
Exited            10000 non-null int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
(10000, 14)
```

3.3 CHECKING NULL VALUES

A null value is a value that has no value exists for the particular position. Dataset with null values affect the performance of the Machine Learning model. Null values in the dataset can be identified and can be eradicated.

```
1 # Check if we have any NaN values
2 customer_data.isnull().values.any()
```

```
False
```

3.4 DESCRIPTIVE ANALYTICS FOR DATA

Descriptive analytics gives you a general view of the historic data, to provide a clear, straightforward picture of the company's operations. Descriptive analytics is the interpretation of historical data to better understand changes that have occurred in a business. Descriptive analytics describes the use of a range of historic data to draw comparisons.

3.4.1 PRECISION SETUP

We can set the options in the pandas setup to make the precision to a number of decimal points that we desire. The `describe()` function is used to get the descriptive statistics of the dataset. The output of this function will be the following:

- count
- mean
- standard deviation
- minimum
- maximum
- 25%, 50%, 75% values

for each attributes of the dataset.

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.000	10000.000	10000.000	10000.000	10000.000	10000.000	10000.000	10000.000	10000.000
mean	650.529	38.922	5.013	76485.889	1.530	0.706	0.515	100090.240	0.204
std	96.653	10.488	2.892	62397.405	0.582	0.456	0.500	57510.493	0.403
min	350.000	18.000	0.000	0.000	1.000	0.000	0.000	11.580	0.000
25%	584.000	32.000	3.000	0.000	1.000	0.000	0.000	51002.110	0.000
50%	652.000	37.000	5.000	97198.540	1.000	1.000	1.000	100193.915	0.000
75%	718.000	44.000	7.000	127644.240	2.000	1.000	1.000	149388.247	0.000
max	850.000	92.000	10.000	250898.090	4.000	1.000	1.000	199992.480	1.000

3.4.2 IDENTIFYING CORRELATIONS

Correlation is a statistical measure that expresses the extent to which two variables are linearly related (meaning they change together at a constant rate). Correlation is a measure of the strength of a linear relationship between two quantitative variables.

The Pearson coefficient is a type of correlation coefficient that represents the relationship between two variables that are measured on the same interval or ratio scale. The Pearson coefficient is a measure of the strength of the association between two continuous variables.

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

r = correlation coefficient

x_i = values of the x-variable in a sample

\bar{x} = mean of the values of the x-variable

y_i = values of the y-variable in a sample

\bar{y} = mean of the values of the y-variable

The results of applying Pearson Correlation to our dataset is as follows:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
CreditScore	1.000e+00	-0.004	8.419e-04	0.006	0.012	-0.005	0.026	-0.001	-0.027
Age	-3.965e-03	1.000	-9.997e-03	0.028	-0.031	-0.012	0.085	-0.007	0.285
Tenure	8.419e-04	-0.010	1.000e+00	-0.012	0.013	0.023	-0.028	0.008	-0.014
Balance	6.268e-03	0.028	-1.225e-02	1.000	-0.304	-0.015	-0.010	0.013	0.119
NumOfProducts	1.224e-02	-0.031	1.344e-02	-0.304	1.000	0.003	0.010	0.014	-0.048
HasCrCard	-5.458e-03	-0.012	2.258e-02	-0.015	0.003	1.000	-0.012	-0.010	-0.007
IsActiveMember	2.565e-02	0.085	-2.836e-02	-0.010	0.010	-0.012	1.000	-0.011	-0.156
EstimatedSalary	-1.384e-03	-0.007	7.784e-03	0.013	0.014	-0.010	-0.011	1.000	0.012
Exited	-2.709e-02	0.285	-1.400e-02	0.119	-0.048	-0.007	-0.156	0.012	1.000

3.5 DATA VISUALIZATION

Data visualization is the discipline of trying to understand data by placing it in a visual context so that patterns, trends and correlations that might not otherwise be detected can be exposed. Python offers multiple great graphing libraries that come packed with lots of different features. Data visualization is the graphical representation of data in order to interactively and efficiently convey insights to clients, customers, and stakeholders in general.

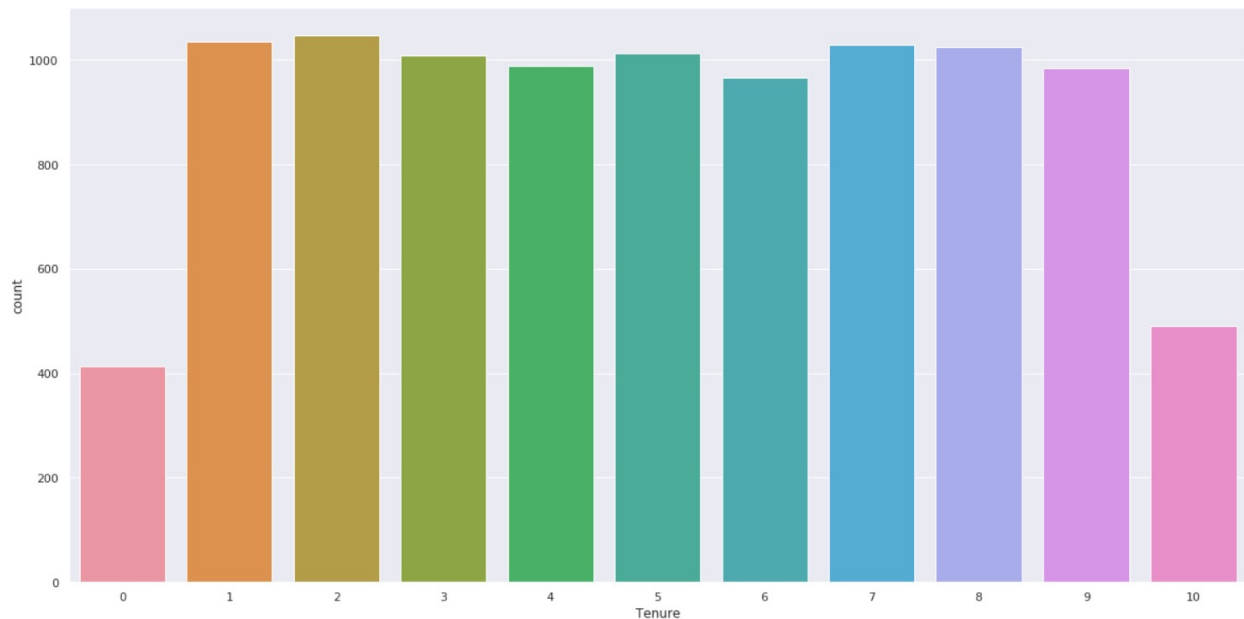
The different types of data visualization techniques used in analysing the dataset are as follows:

3.5.1 COUNT PLOTS

The `countplot()` method is used to show the counts of observations in each categorical bin using bars. The `countplot()` is applied for the following attributes of the dataset:

- Tenure
- Credit Score
- Geography

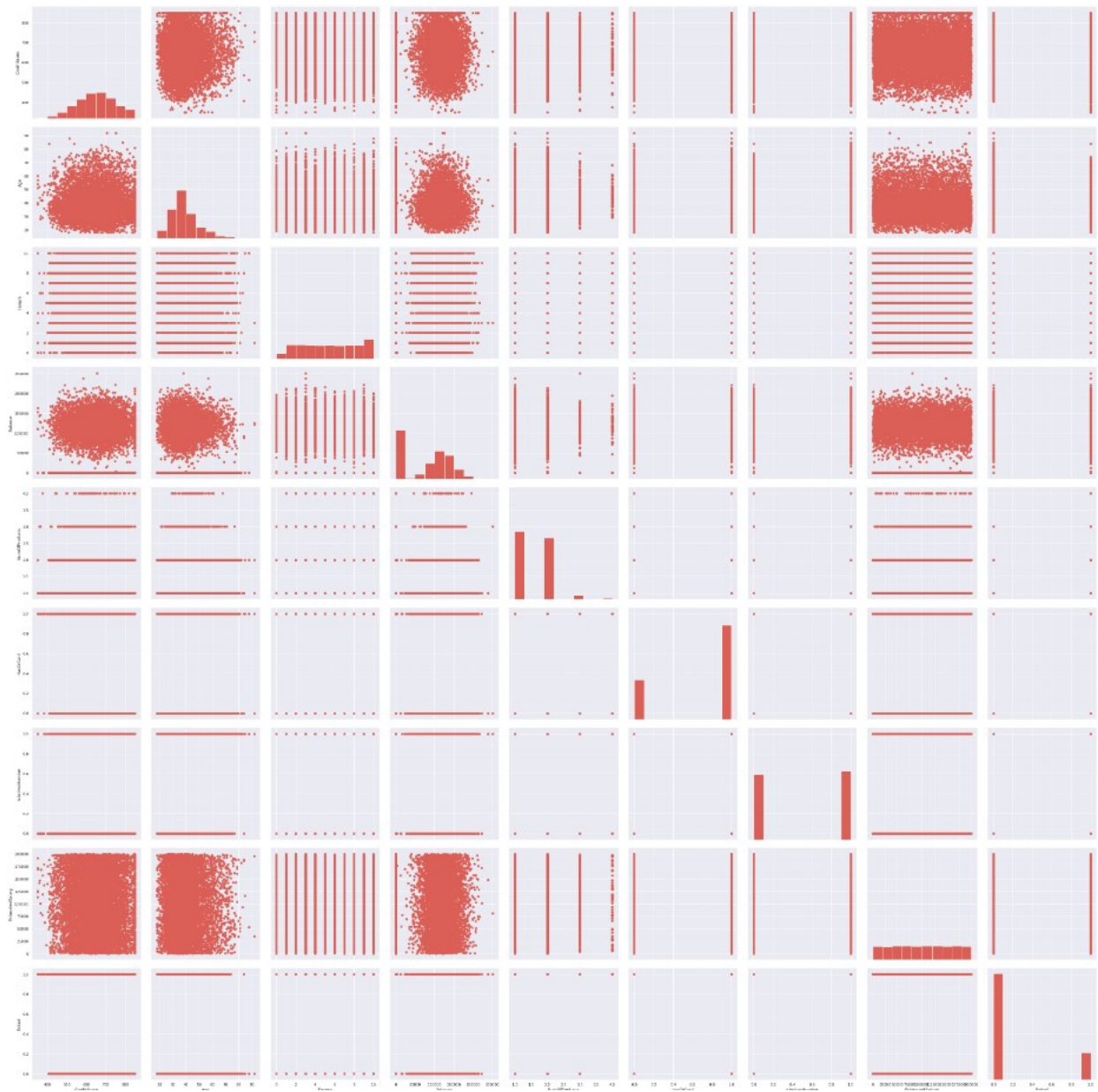
The output of a countplot is as follows:



3.5.2 PAIRGRID GENERATION

A pairgrid is a subplot grid for plotting pairwise relationships in a dataset. This object maps each variable in a dataset onto a column and row in a grid of multiple axes. Different axes-level plotting functions can be used to draw bivariate plots in the upper and lower triangles, and the marginal distribution of each variable can be shown on the diagonal.

The pairgrid shows the relationship between an attribute of the dataset with any other attribute of the dataset. The dense spots indicate the strong relationships between the attributes. The sparse spots indicate the weak relationships between the attributes. The pairgrid that is generated for the dataset under consideration is as follows:



3.6 DATA DISTRIBUTION

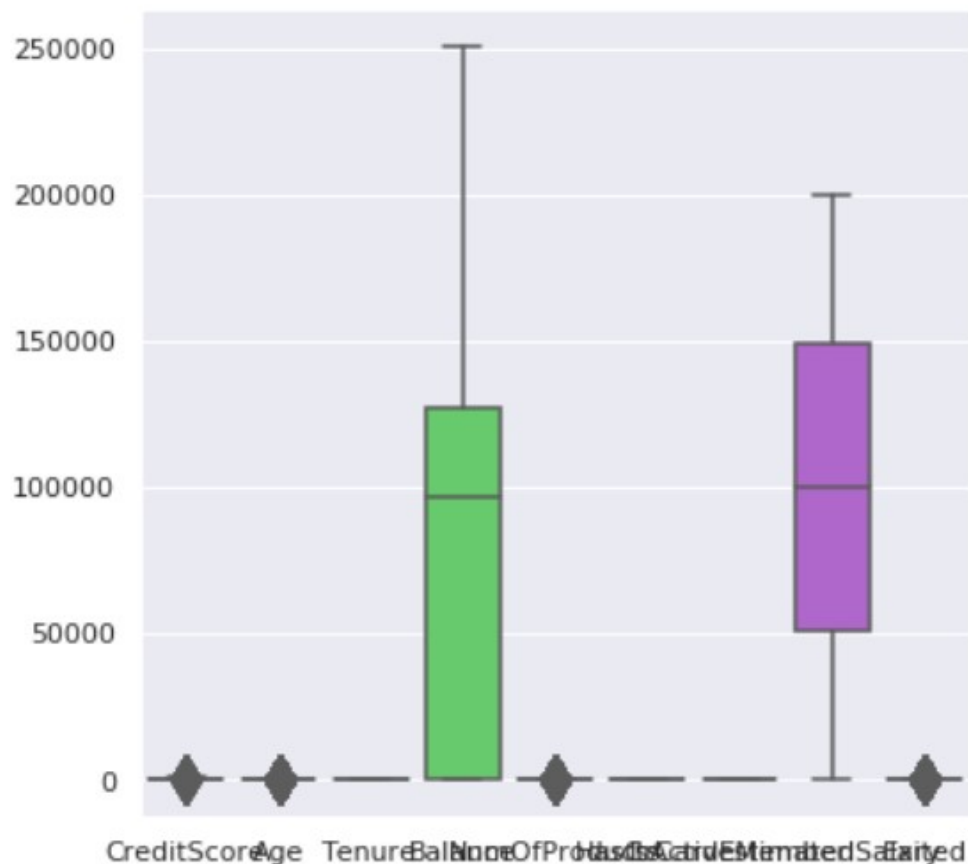
A data distribution is a function or a listing which shows all the possible values (or intervals) of the data. It also (and this is important) tells you how often each value occurs. Often, the data in a distribution will be ordered from smallest to largest, and graphs and charts allow you to easily see both the values and the frequency with which they appear.

From a distribution you can calculate the probability of any one particular observation in the sample space, or the likelihood that an

observation will have a value which is less than (or greater than) a point of interest. The function of a distribution that shows the density of the values of our data is called a probability density function, and is sometimes abbreviated pdf. The methods of data distributions used in the project are as follows:

3.6.1 BOXPLOTS

A box plot (or box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable. The box shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution, except for points that are determined to be “outliers” using a method that is a function of the inter-quartile range. The boxplot for the dataset is as follows:

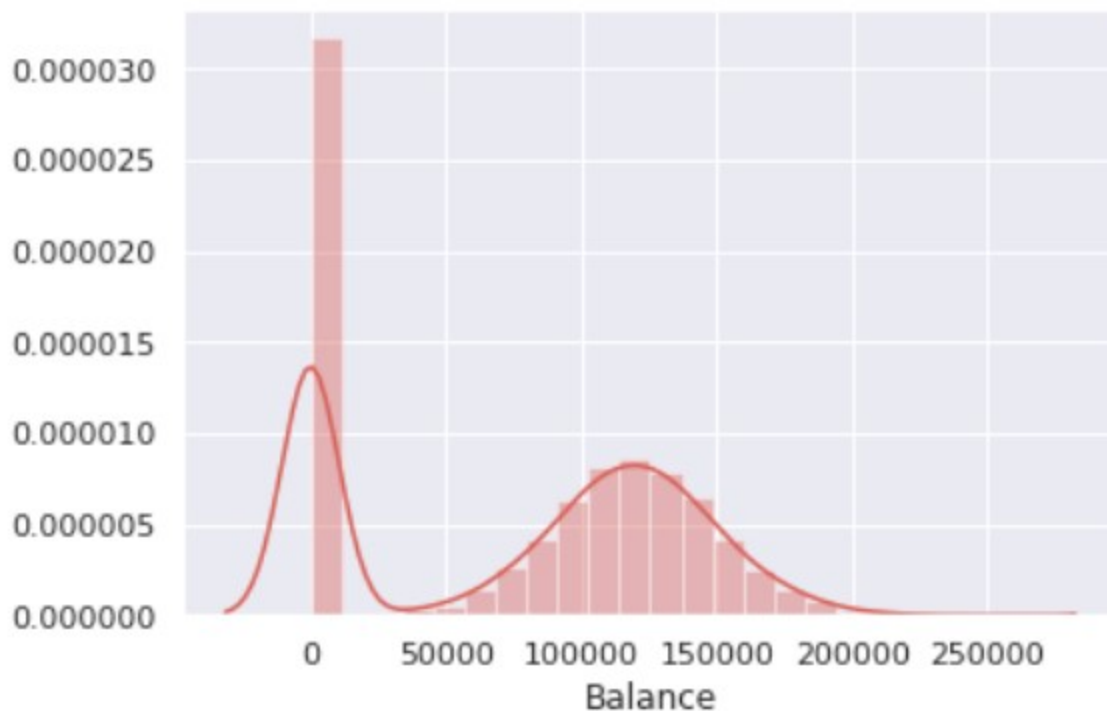


3.6.2 DISTRIBUTION PLOTS

Seaborn distplot lets you show a histogram with a line on it. We use seaborn in combination with matplotlib, the Python plotting module. A distplot plots a univariate distribution of observations. The distplot() function combines the matplotlib hist function with the seaborn kdeplot() and rugplot() functions.

The histogram shows buckets of data ranges called as bins and distributes the values of the attributes into the buckets. Then, it calculates the probability of occurrence of each of the buckets. This process is done for categorical data. For continuous data, a PDF curve is generated, which shows the distribution of categorical data as a function of a polynomial, which generates a curve.

A sample output for a distplot is as follows:



The distplots are generated for the following attributes of the dataset:

- Balance
- # of Products
- Estimated Salary

3.7 ONE-HOT ENCODING

Sometimes in datasets, we encounter columns that contain numbers of no specific order of preference. The data in the column usually denotes a category or value of the category and also when the data in the column is label encoded. This confuses the machine learning model, to avoid this the data in the column should be One Hot encoded. One-hot encoding refers to splitting the column which contains numerical categorical data to many columns depending on the number of categories present in that column. Each column contains “0” or “1” corresponding to which column it has been placed.

For non pre-processed data, `LabelEncoder()` helps to generate one-hot encoding for the dataset. For pre-processed data, `pd.get_dummies()` function helps to generate one-hot encoding for the dataset.

An example for one-hot encoding is as follows:

Color		Red	Yellow	Green
Red		1	0	0
Red		1	0	0
Yellow		0	1	0
Green		0	0	1
Yellow				

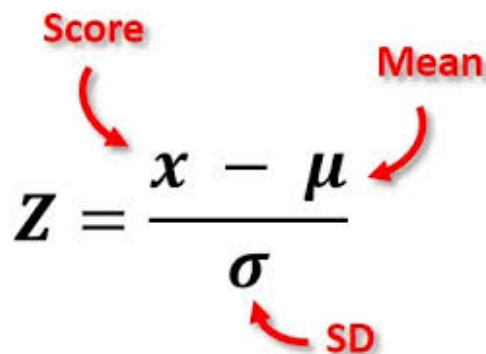
3.8 OUTLIER DETECTION

The presence of outliers in a classification or regression dataset can result in a poor fit and lower predictive modeling performance. Identifying and removing outliers is challenging with simple statistical methods for most machine learning datasets given the large number of input variables. Instead, automatic outlier detection methods can be used in the modeling pipeline and compared, just like other data preparation transforms that may be applied to the dataset.

The methods used for outlier detection are as follows:

3.8.1 Z-SCORE

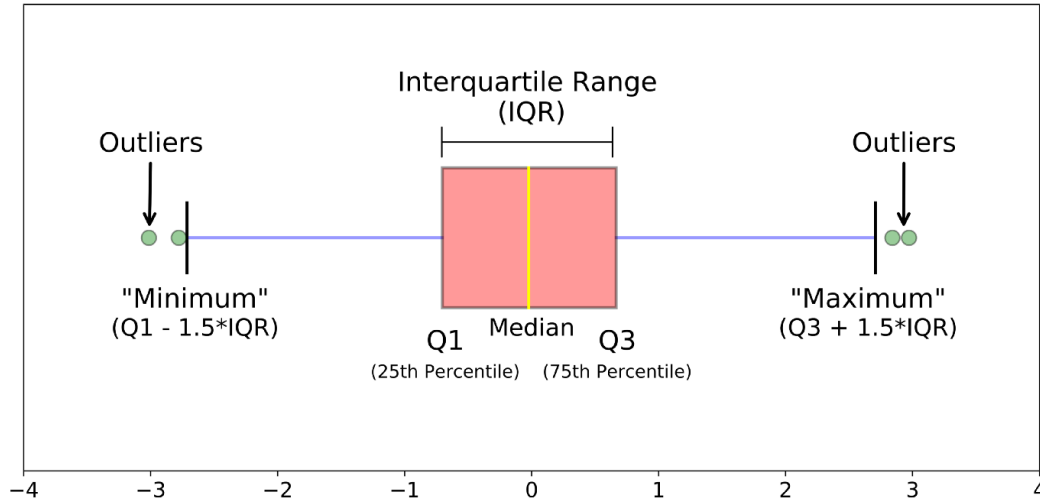
Z score is an important concept in statistics. Z score is also called standard score. This score helps to understand if a data value is greater or smaller than mean and how far away it is from the mean. More specifically, Z score tells how many standard deviations away a data point is from the mean. The Z-Score method is applied to 'EstimatedSalary' attribute, and it showed no presence of an outlier. The formula for Z-Score is as follows:

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


3.8.2 INTER-QUARTILE RANGE METHOD

In descriptive statistics, the interquartile range, also called the midspread or middle 50%, or technically H-spread, is a measure of statistical dispersion, being equal to the difference between 75th and 25th percentiles, or between upper and lower quartiles, $IQR = Q_3 - Q_1$. In our dataset, the IQR (Inter-Quartile Range) method is applied to 'Ballance' and it showed no evidence of outliers.

The formula for IQR method is as follows:



3.9 FEATURE ENGINEERING

All machine learning algorithms use some input data to create outputs. This input data comprise features, which are usually in the form of structured columns. Algorithms require features with some specific characteristic to work properly. Here, the need for feature engineering arises. The features we use influence more than everything else the result. No algorithm alone can supplement the information gain given by correct feature engineering. The method of feature engineering that is used in our project is "Polynomial Features"

3.9.1 POLYNOMIAL FEATURES

Polynomial features are those features created by raising existing features to an exponent. For example, if a dataset had one input feature X , then a polynomial feature would be the addition of a new feature (column) where values were calculated by squaring the values in X , e.g. X^2 . This process can be repeated for each input variable in the dataset, creating a transformed version of each. As such, polynomial features are a type of feature engineering, e.g. the creation of new input features based on the existing features. The "degree" of the polynomial is used to control the

number of features added, e.g. a degree of 3 will add two new variables for each input variable. Typically a small degree is used such as 2 or 3.

The output of the feature engineering had 87 attributes in our dataset, and among them, the best 25 were selected.

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Geography_France	Geography_Germany	Geography_Spain
0	619.0	42.0	2.0	0.00	1.0	1.0	1.0	101348.88	1.0	0.0	0.0
1	608.0	41.0	1.0	83807.86	1.0	0.0	1.0	112542.58	0.0	0.0	1.0
2	502.0	42.0	8.0	159660.80	3.0	1.0	0.0	113931.57	1.0	0.0	0.0
3	699.0	39.0	1.0	0.00	2.0	0.0	0.0	93826.63	1.0	0.0	0.0
4	850.0	43.0	2.0	125510.82	1.0	1.0	1.0	79084.10	0.0	0.0	1.0
5	645.0	44.0	8.0	113755.78	2.0	1.0	0.0	149756.71	0.0	0.0	1.0
6	822.0	50.0	7.0	0.00	2.0	1.0	1.0	10062.80	1.0	0.0	0.0
7	376.0	29.0	4.0	115046.74	4.0	1.0	0.0	119346.88	0.0	1.0	0.0
8	501.0	44.0	4.0	142051.07	2.0	0.0	1.0	74940.50	1.0	0.0	0.0
9	684.0	27.0	2.0	134603.88	1.0	1.0	1.0	71725.73	1.0	0.0	0.0
10	528.0	31.0	6.0	102016.72	2.0	0.0	0.0	80181.12	1.0	0.0	0.0
11	497.0	24.0	3.0	0.00	2.0	1.0	0.0	76390.01	0.0	0.0	1.0
12	476.0	34.0	10.0	0.00	2.0	1.0	0.0	26260.98	1.0	0.0	0.0
13	549.0	25.0	5.0	0.00	2.0	0.0	0.0	190857.79	1.0	0.0	0.0
14	635.0	35.0	7.0	0.00	2.0	1.0	1.0	65951.65	0.0	0.0	1.0

15 rows × 87 columns

3.10 FEATURE SCALING

Feature scaling is a method used to normalize the range of independent variables or features of data. Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step.

For every feature, the minimum value of that feature gets transformed into a 0, the maximum value gets transformed into a 1, and every other value gets transformed into a decimal between 0 and 1. Min-max normalization has one fairly significant downside: it does not handle outliers very well.

The output for scaling the independent attributes is as follows:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Geography_France	Geography_Germany	Geography_Spain
0	1.676	-0.942	0.009	0.479	-0.905	-1.546	-1.036	1.505	-1.004	1.749	-0.58
1	0.303	-0.180	1.047	-0.047	0.820	0.647	0.966	-1.312	-1.004	1.749	-0.58
2	-0.102	0.963	-0.337	0.712	-0.905	0.647	-1.036	-1.321	-1.004	1.749	-0.58
3	-0.414	-0.751	1.047	0.669	-0.905	0.647	0.966	-0.812	0.996	-0.572	-0.58
4	-0.685	-0.561	1.393	-1.216	-0.905	0.647	-1.036	-1.629	0.996	-0.572	-0.58

3.11 CREATING TRAIN AND TEST DATA

The training data and testing data are created from the pre-processed dataset. The function of the training data is to train the model and improve its understanding about the dataset and its attributes, across many epochs and batches. The function of the test data is to evaluate the model's understanding to the problem.

In this project, we have splitted the training and test data in the ratio of 2:1. The output of the shape of the train and test data is as follows:

```
(6700, 10) (6700,)
(3300, 10) (3300,)
```

3.12 MODEL CREATION

The Machine Learning model is created by invoking appropriate functions that are available in "scikit-learn" package in Python. There are various parameters which can be used under different scenarios for creating the Machine Learning model. In our project, there are 4 different models taken into consideration. They are as follows:

- Support Vector Classifier on non pre-processed data
- Support Vector Classifier on pre-processed data
- Logistic Regression
- Multi Layer Perceptron (Neural Network)

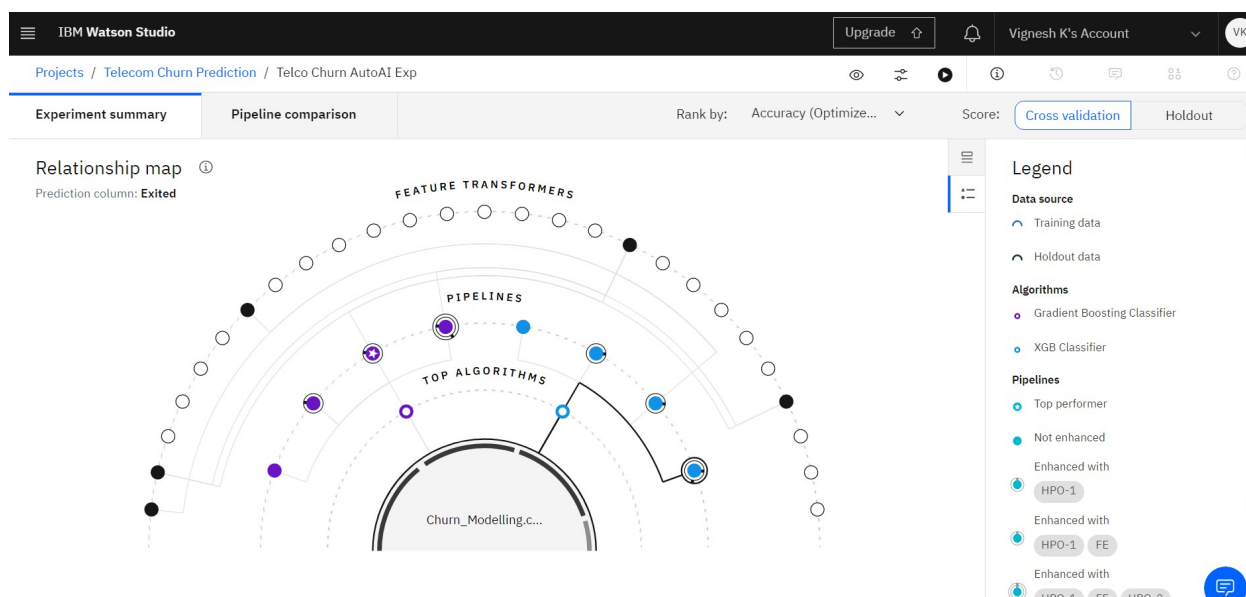
We can get the description of the model's parametrs once we fit the model with the training and test data. The sample output for creating an ML model is as follows:

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='rbf', max_iter=-1, probability=False, random_state=42,
    shrinking=True, tol=0.001, verbose=False)
```

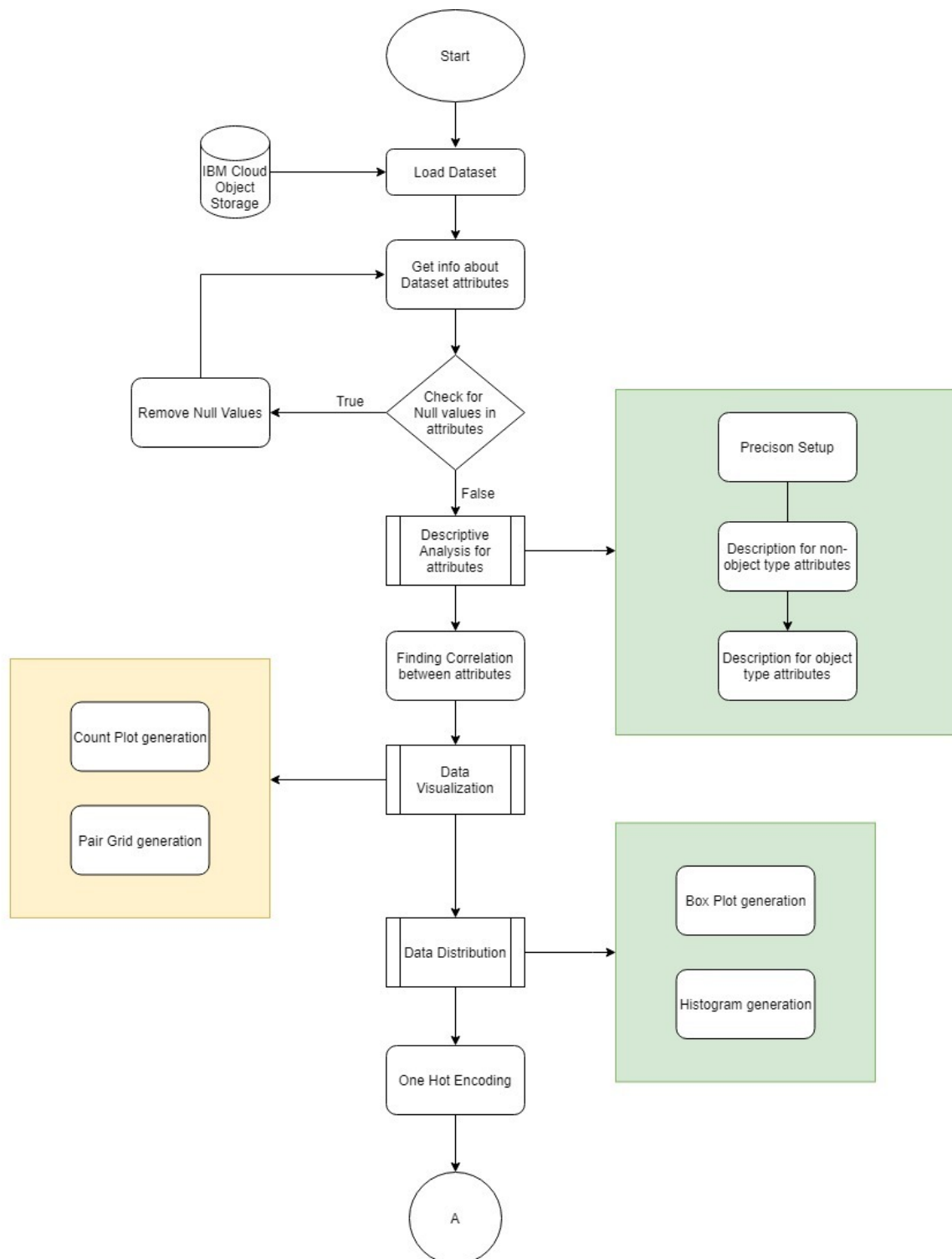
3.13 AUTO AI

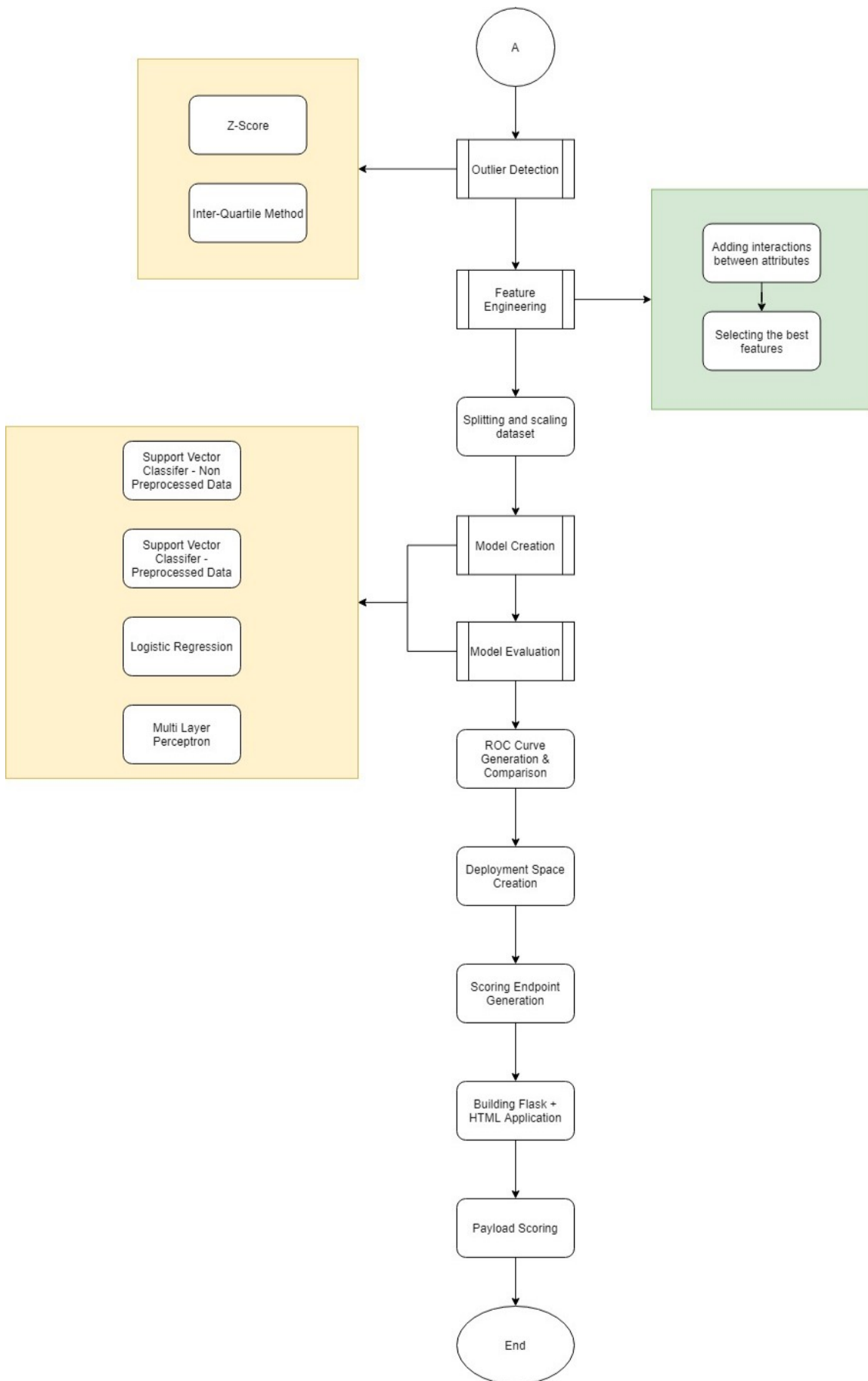
The AutoAI graphical tool in Watson Studio automatically analyzes data and generates candidate model pipelines customized for predictive modeling problems. These model pipelines are created iteratively as AutoAI analyzes your dataset and discovers data transformations, algorithms, and parameter settings that work best for problem setting. Results are displayed on a leaderboard, showing the automatically generated model pipelines ranked according to problem optimization objective. AutoAI enables AI and ML end-to-end lifecycle management.

The result of AutoAI in our dataset is as follows:



FLOW CHART





The flowchart depicts the sequential implementation of the proposed system. The flowchart shows the dependencies, the independent and dependent tasks. The flowchart helps to organize the system functionalities.

Here, the proposed system is implemented by creating a Watson Machine Learning instance. Here, the use of a python notebook is essential as it is helpful for processes like dataset preparation, data pre-processing, data visualization, feature engineering, model creation, and model prediction.

Once the Machine Learning model is ready-to-use, the deployment space is created, in which the deployment model is created and added to the deployment assets. When the asset is deployed successfully, the scoring endpoint is generated.

The flask application process consists of HTML form creation, flask integration, and scoring endpoint integration. Once these processes are done, the application can be executed and the ML model automation will be complete.

RESULTS

The results that has been obtained as a result of evaluating the ML models that are created are listed below. Also, a comparison has been made on which algorithm (model) works better for the dataset provided. The deployment process of the model and scoring endpoint automation is also mentioned here.

6.1 MODEL EVALUATION

Model evaluation aims to estimate the generalization accuracy of a model on future (unseen/out-of-sample) data. It helps to find the best model that represents our data and how well the chosen model will work in the future. Model evaluation metrics are used to assess goodness of fit between model and data, to compare different models, in the context of model selection, and to predict how predictions (associated with a specific model and data set) are expected to be accurate. The three main metrics used to evaluate a classification model are accuracy, precision, and recall.

Accuracy is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions.

$$\text{accuracy} = \frac{\text{correct predictions}}{\text{all predictions}}$$

Precision is defined as the fraction of relevant examples (true positives) among all of the examples which were predicted to belong in a certain class.

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

Recall is defined as the fraction of examples which were predicted to belong to a class with respect to all of the examples that truly belong in the class.

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

The sample output of obtaining the metrics is given as follows:

```
1 # Get model confidence of predictions
2 y_score_svc = clf_svc.decision_function(X_test_scaled)
3 y_score_svc

array([-1.35894336, -1.13050684, -1.01055346, ..., -1.30490437,
       -0.8450336 ,  0.98776799])

1 # Get accuracy score
2 y_pred_svc = clf_svc.predict(X_test_scaled)
3 acc_svc = accuracy_score(y_test, y_pred_svc)
4 print(acc_svc)

0.8615151515151516

1 # Get Precision vs. Recall score
2 average_precision_svc = average_precision_score(y_test, y_score_svc)
3
4 print('Average precision-recall score: {0:0.2f}'.format(
5     average_precision_svc))

Average precision-recall score: 0.66
```

The following table shows the readings of the metrics as the output of model evaluation:

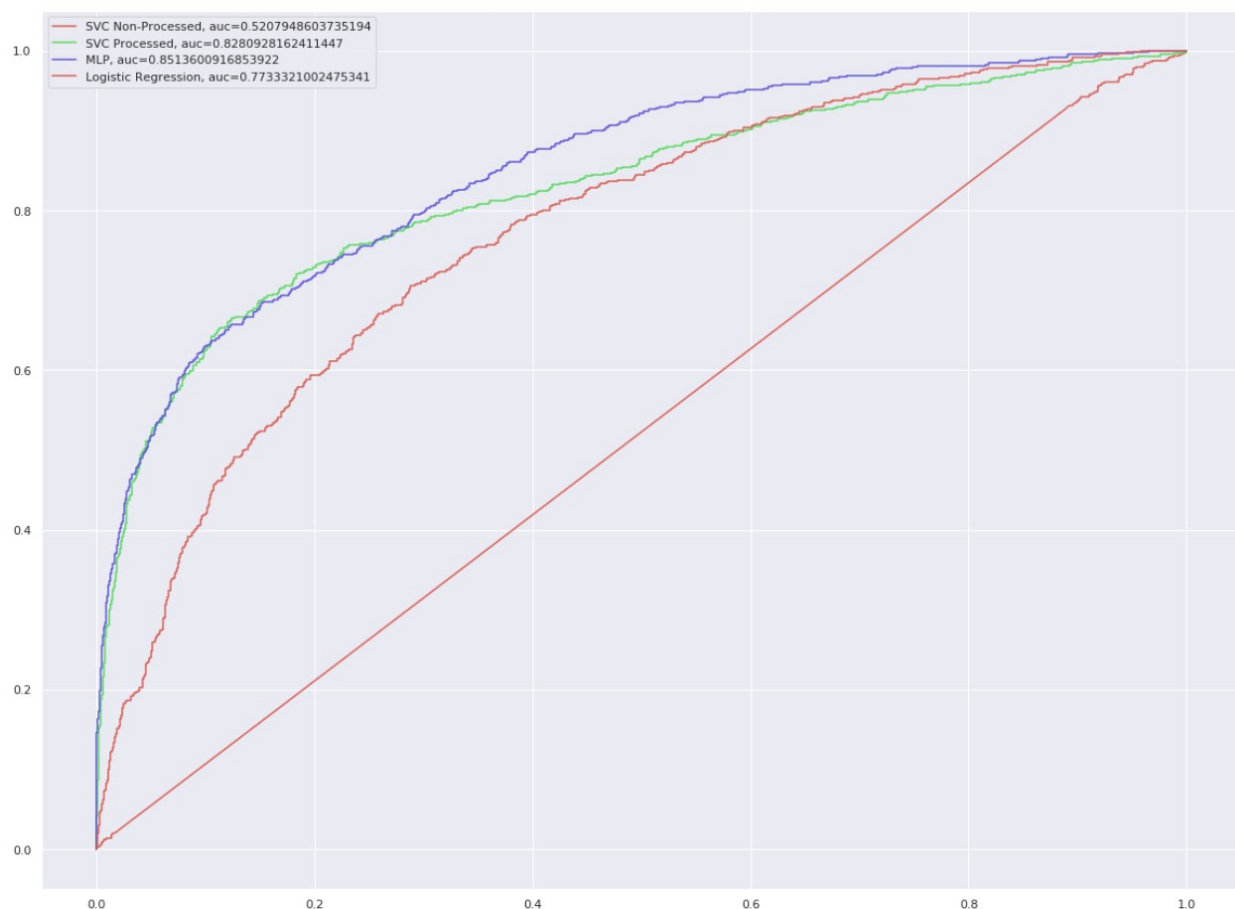
ALGORITHM	ACCURACY SCORE	PRECISION - RECALL SCORE
Support Vector Classifier on non pre-processed data	0.8051	0.20
Support Vector Classifier on pre-processed data	0.8615	0.66
Logistic Regression	0.8115	0.46
Multi Layer Perceptron	0.8651	0.68

6.2 ROC CURVE GENERATION

A receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. An ROC curve is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: True Positive Rate and False Positive Rate.

ROC curves are frequently used to show in a graphical way the connection/trade-off between clinical sensitivity and specificity for every possible cut-off for a test or a combination of tests. In addition the area under the ROC curve gives an idea about the benefit of using the test(s) in question.

The ROC curve generated for the ML models is shown below:



The curve shows that MLP and SVC pre-processed show better results. The Logistic Regression model shows an average performance and SVC non pre-processed has a low TPR due to the nature of the data fed into the model.

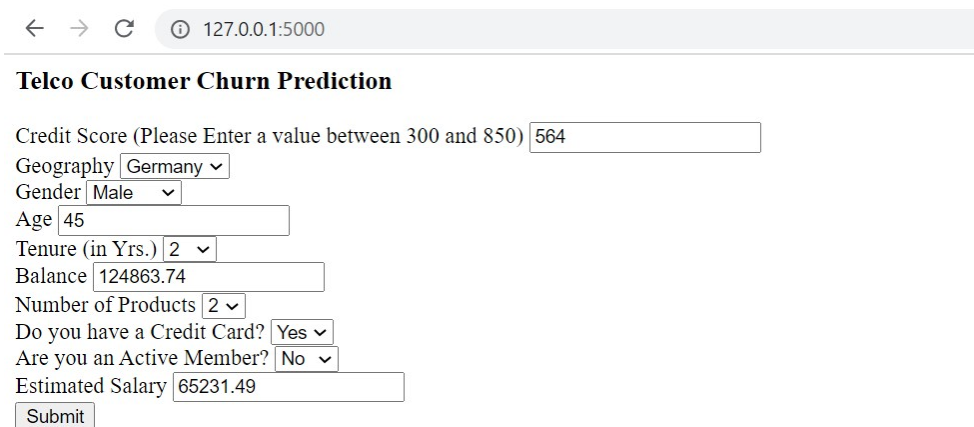
6.3 BUILDING FLASK APPLICATION

IBM provides the option of deploying the ML models created in Watson Studio to get deployed in real time by providing dynamic scoring endpoint URLs. This enables users to create models and deploy them effectively.

The scoring endpoint URL can be obtained by creating a deployment model and adding it to the deployment space as an instance. This enables multiple model to get deployed simultaneously. I have deployed the model "Support Vector Classifier on pre-processed data" into a deployment space.

The scoring endpoint URL obtained for by deploying the model is: <https://eu-gb.ml.cloud.ibm.com/ml/v4/deployments/a77fd05b-67a5-40d1-8ab1-17e160b261c8/predictions?version=2020-10-20>

A flask application is built in order to perform automated deployment of the ML model. The UI is built using HTML by creating a form to get the independent variables of the dataset as the user inputs. The output of the UI is given below:



← → ↻ ⓘ 127.0.0.1:5000

Telco Customer Churn Prediction

Credit Score (Please Enter a value between 300 and 850)

Geography

Gender

Age

Tenure (in Yrs.)

Balance

Number of Products

Do you have a Credit Card?

Are you an Active Member?

Estimated Salary

After the UI is built, the python script is built and executed. The execution of the script makes the flask app to get deployed onto the local server (http://127.0.0.1:5000) port number 5000.

 C:\Windows\System32\cmd.exe - python script.py

```
C:\Users\Palani\churn_prediction>python script.py
* Serving Flask app "script" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Once the user clicks the "Submit" button, the responses get recoded and the independent attributes of the dataset are transformed into the pattern into which it is sent into the ML model. The payload is created in the pattern of "[fields]:[values]" and is sent along with the URL as a POST request. The model present in the deployment makes the prediction and sends the result back to the local server in JSON format. The prediction of the model is printed in the result page.

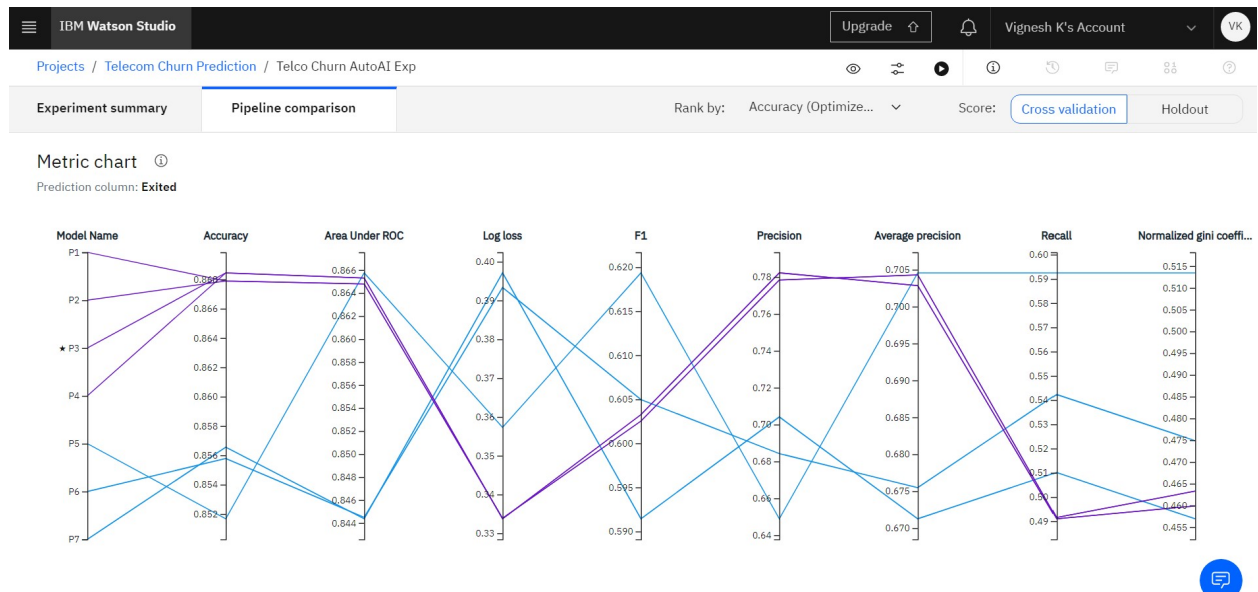
    127.0.0.1:5000/result

The customer will exit from the company

6.4 AUTO AI PIPELINE DETAILS

The AutoAI experiment is run on IBM Watson Studio by feeding it with the dataset. A pipeline is generated which had two different algorithms, with different versions, by varying critical parameters of Machine Learning. All the models created and run automatically and the results are provided with ample amount of metrics available for comparison.

The result of pipeline comparison is shown below:



The final list with the AutoAI algorithms into consideration along with the computed metrics is given below:

Rank	↑	Name	Algorithm	Accuracy (Optimized)	Average pre...	F1	Log loss	Normalized ...	Precision	Recall
★ 1		Pipeline 3	Gradient Boosting Classif...	0.868	0.703	0.603	0.334	0.460	0.782	0.491
2		Pipeline 4	Gradient Boosting Classif...	0.868	0.703	0.603	0.334	0.460	0.782	0.491
3		Pipeline 1	Gradient Boosting Classif...	0.868	0.704	0.603	0.334	0.463	0.778	0.492
4		Pipeline 2	Gradient Boosting Classif...	0.868	0.704	0.603	0.334	0.463	0.778	0.492
5		Pipeline 7	XGB Classifier	0.857	0.671	0.591	0.397	0.457	0.704	0.510
6		Pipeline 6	XGB Classifier	0.856	0.675	0.605	0.393	0.475	0.684	0.542
7		Pipeline 5	XGB Classifier	0.852	0.705	0.619	0.357	0.513	0.649	0.592

ADVANTAGES & DISADVANTAGES

7.1 ADVANTAGES

- The services provided by IBM Cloud can be leveraged to perform complex tasks of any scale with ease.
- The interface is easy to use, with the tours guiding through all the important aspects of the services.
- Usage of python is easy and handy when it comes to data visualization and analysing data distribution.
- Access to a wide range of software assets which can be incorporated into the project in just a few clicks.
- Access to Auto AI has made a huge impact on the project. It enables even naive users to understand Machine Learning algorithms and many more techniques.
- Production deployments and automation using payload scoring gives exposure of handling end-to-end application.

7.2 DISADVANTAGES

- Exceeding capacity-units per hour (CUH) imposes a bottleneck in utilizing the capabilities.

APPLICATIONS

The proposed system will be helpful in predicting whether a customer will leave the company or continue with the company. This will be beneficial for Telecom companies who suffer significant losses due to customer churn. The rate at which customers churn from a company is called as churn rate. The churn prediction will try to reduce the churn rate as minimum as possible, playing an important role in the company's turnover and reputation.

The use of an application to make adhoc predictions will help the users of the application to get instant results for the inputs. The parameters chosen for predicting customer churn are spot-on, and all of them are very critical for predicting the churn rate.

The use of churn prediction beforehand will enable the company to make counter attacks and try to retain more customers by introducing optimized plans, new offers etc., It also helps the company to avoid unnecessary loss and also adds up new customers due to improvised workflow strategies.

CONCLUSION

I would like to extend my gratitude to IBM India Pvt. Ltd. and SmartInternz - by SmartBridge Educational Services, for giving me an opportunity to use the resources, study materials, tutorials provided by them and to have me as a part of IBM Build-a-thon.

I have built a project named "Telecom Churn Prediction using Watson Auto AI" and have been provided free Watson Studio Desktop access for 30 days. I think I have done justice for the opportunity and the resources provided to me.

It has been an enthralling experience for me working under this project for 3 weeks. I have recorded a video to demonstrate the working of the project. I have also added all the resources from my side to the Git.

Scoring Endpoint URL:

<https://eu-gb.ml.cloud.ibm.com/ml/v4/deployments/a77fd05b-67a5-40d1-8ab1-17e160b261c8/predictions?version=2020-10-20>

GitHub link to my project:

<https://github.com/SmartPracticeschool/SPS-5382-Telecom-Customer-Churn-Prediction-using-Watson-Auto-AI>

Link to the Project Demonstration Video:

<https://drive.google.com/file/d/1zsHleclcB76JRT8yOepPMIURsCig0nYi/view?usp=sharing>

FUTURE SCOPE

The project can be enhanced from different view points namely:

- Optimized Machine Learning algorithms
- More feature engineering techniques
- Analysing vital parameters for targeted customers
- Flask UI with improved functionalities
- Multiple deployments for different business scenarios

BIBLIOGRAPHY

1. *Essam Shaaban, Yehia Helmy, Ayman Khedr, Mona Nasr* | **International Journal of Engineering Research and Applications (IJERA)** | A Proposed Churn Prediction Model
2. *Sandra Mitrović, Bart Baesens, Wilfried Lemahieu, Jochen De Weerd* | On the Operational Efficiency of Different Feature Types for Telco Churn Prediction
3. *Veronikha Effendy, Adiwijaya, Z.K.A. Baizal.* | **2014 2nd International Conference on Information and Communication Technology (ICoICT)** | Handling Imbalanced Data in Customer Churn Prediction Using Combined Sampling and Weighted Random Forest.
4. *Yiqing Huang, Fangzhou Zhu, Mingxuan Yuan, Ke Deng, Yanhua Li, Bing Ni, Wenyuan Dai, Qiang Yang, Jia Zeng* | **Advancing Computing as a Science & Profession** | Telco Churn Prediction with Big Data.
5. *Sandra Mitrović, Bart Baesens, Wilfried Lemahieu, Jochen De Weerd* | Churn Prediction using Dynamic RFM-Augmented node2vec.

APPENDIX

A. SOURCE CODE

```
"""# Customer Churn Prediction
```

```
## 1. Loading Libraries
```

```
"""
```

```
import json
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from itertools import combinations
from sklearn.preprocessing import PolynomialFeatures, LabelEncoder,
StandardScaler
import sklearn.feature_selection
from sklearn.model_selection import train_test_split
from collections import defaultdict
from sklearn import metrics
import pickle
```

```
"""### The Dataset
```

From a telecommunications company. It includes information about:

- Customers who left within the last month – the column is called Churn
- Services that each customer has signed up for – phone, multiple lines,

internet, online security, online backup, device protection, tech support, and streaming TV and movies

- Customer account information – how long they've been a customer, contract, payment method, paperless billing, monthly charges, and total charges

- Demographic info about customers – gender, age range, and if they have partners and dependents

2. Loading Our Dataset

Click on the cell below to highlight it.

Then go to the `Files` section to the right of this notebook and click `Insert to code` for the data you have uploaded. Choose `Insert pandas DataFrame`.

```
"""
```

```
import types
import pandas as pd
from botocore.client import Config
import ibm_boto3
```

```
def __iter__(self): return 0
```

```
# @hidden_cell
```

```
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
```

```
# You might want to remove those credentials before you share the notebook.
```

```
client_b874c30054d441ffacbe02cbcc8859e6 =
```

```
ibm_boto3.client(service_name='s3',
                  ibm_api_key_id='***',
                  ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
                  config=Config(signature_version='oauth'),
                  endpoint_url='https://s3.eu-geo.objectstorage.service.networklayer.com')
```

```
body =
client_b874c30054d441ffacbe02cbcc8859e6.get_object(Bucket='telcochurnpredicition-donotdelete-pr-2use5r9izvml7k',Key='Churn_Modelling.csv')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__,
body )
```

```
df_data_2 = pd.read_csv(body)
df_data_2.head()
```

```
customer_data = df_data_2
```

```
# Checking that everything is correct
pd.set_option('display.max_columns', 30)
customer_data.head(10)
```

```
"""### 3. Get some info about our Dataset and whether we have missing values"""
```

```
# After running this cell we will see that we have no missing values
customer_data.info()
customer_data.shape
```

```
# Drop customerID column
customer_data = customer_data.drop('RowNumber', axis=1)
```

```
customer_data = customer_data.drop('CustomerId', axis=1)
customer_data = customer_data.drop('Surname', axis=1)
customer_data.head(5)
```

```
# Check if we have any NaN values
customer_data.isnull().values.any()
```

```
customer_data.info()
```

```
"""### 4. Descriptive analytics for our data"""
```

```
# Describe columns with numerical values
pd.set_option('precision', 3)
customer_data.describe()
```

```
# Describe columns with objects
customer_data.describe(exclude=np.number)
```

```
# Find correlations
customer_data.corr(method='pearson')
```

```
"""### 5. Visualize our Data to understand it better
```

```
#### Plot Relationships
"""
```

```
# Plot Tenure Frequency count
sns.set(style="darkgrid")
sns.set_palette("hls", 3)
fig, ax = plt.subplots(figsize=(20,10))
ax = sns.countplot(x="Tenure", data=customer_data)
```

```
# Plot CreditScore Frequency count
sns.set(style="darkgrid")
sns.set_palette("hls", 3)
fig, ax = plt.subplots(figsize=(20,10))
ax = sns.countplot(x="CreditScore", data=customer_data)
```

```
# Plot Geography Frequency count
sns.set(style="darkgrid")
sns.set_palette("hls", 3)
fig, ax = plt.subplots(figsize=(20,10))
ax = sns.countplot(x="Geography", data=customer_data)
```

```
# Create Grid for pairwise relationships
gr = sns.PairGrid(customer_data, size=5)
gr = gr.map_diag(plt.hist)
gr = gr.map_offdiag(plt.scatter)
gr = gr.add_legend()
```

```
"""#### Understand Data Distribution"""
```

```
# Set up plot size
fig, ax = plt.subplots(figsize=(6,6))
```

```
# Attributes distribution
a = sns.boxplot(orient="v", palette="hls", data=customer_data.iloc[:,
fliersize=14)
```

```
# Tenure data distribution
histogram = sns.distplot(customer_data.iloc[:, 5], hist=True)
plt.show()
```

```
# Monthly Charges data distribution
```



```
histogram = sns.distplot(customer_data.iloc[:, 6], hist=True)
plt.show()
```

```
# Total Charges data distribution
```

```
histogram = sns.distplot(customer_data.iloc[:, 7], hist=True)
plt.show()
```

```
customer_data1 = customer_data
```

```
customer_data1 = customer_data1.drop('Exited', axis=1)
```

```
customer_data1.head(5)
```

```
"""### 6. Encode string values in data into numerical values"""
```

```
# Use pandas get_dummies
```

```
customer_data_encoded = pd.get_dummies(customer_data1)
```

```
print(customer_data_encoded.head(10))
```

```
customer_data_encoded.shape
```

```
"""### 7. Create Training Set and Labels"""
```

```
# Create training data for non-preprocessed approach
```

```
X_npp = customer_data.iloc[:, :-1].apply(LabelEncoder().fit_transform)
```

```
pd.DataFrame(X_npp).head(5)
```

```
# Create training data for that will undergo preprocessing
```

```
X = customer_data_encoded
```

```
X.head()
```

```
print(X.shape)
```

```
# Extract labels
```

```
y_unenc = customer_data['Exited']
```

```
# Convert strings of 'yes' and 'no' to binary values of 0 or 1
le = preprocessing.LabelEncoder()
le.fit(y_unenc)
```

```
y_le = le.transform(y_unenc)
pd.DataFrame(y_le)
```

```
"""### 8. Detect outliers in numerical values"""
```

```
# Calculate the Z-score using median value and median absolute deviation
for more robust calculations
# Working on EstimatedSalary column
threshold = 3
```

```
median = np.median(X['EstimatedSalary'])
median_absolute_deviation = np.median([np.abs(x - median) for x in
X['EstimatedSalary']])
modified_z_scores = [0.6745 * (x - median) / median_absolute_deviation
                      for x in X['EstimatedSalary']]
results = np.abs(modified_z_scores) > threshold

print(np.any(results))
```

```
# Do the same for Balance column but using the interquartile method
```

```
quartile_1, quartile_3 = np.percentile(X['Balance'], [25, 75])
iqr = quartile_3 - quartile_1
lower_bound = quartile_1 - (iqr * 1.5)
upper_bound = quartile_3 + (iqr * 1.5)

print(np.where((X['Balance'] > upper_bound) | (X['Balance'] < lower_bound)))
```

```
print(X)
X.shape
```

```
# Find interactions between current features and append them to the
dataframe
```

```
def add_interactions(dataset):
```

```
    # Get feature names
```

```
    comb = list(combinations(list(dataset.columns), 2))
```

```
    col_names = list(dataset.columns) + ['_'.join(x) for x in comb]
```

```
    # Find interactions
```

```
    poly = PolynomialFeatures(interaction_only=True, include_bias=False)
```

```
    dataset = poly.fit_transform(dataset)
```

```
    dataset = pd.DataFrame(dataset)
```

```
    dataset.columns = col_names
```

```
    # Remove interactions with 0 values
```

```
    no_inter_indexes = [i for i, x in enumerate(list((dataset == 0).all())) if x]
```

```
    dataset = dataset.drop(dataset.columns[no_inter_indexes], axis=1)
```

```
    return dataset
```

```
print(X)
X.shape
```

```
X_inter = add_interactions(X)
```

```
X_inter.head(15)
```

```
# Select best features
```

```
select = sklearn.feature_selection.SelectKBest(k=25)
```

```
selected_features = select.fit(X_inter, y_le)
```

```
indexes = selected_features.get_support(indices=True)
```

```
col_names_selected = [X_inter.columns[i] for i in indexes]
```

```
X_selected = X_inter[col_names_selected]
```

```
X_selected.head(10)
```

```
"""### 10. Split our dataset into train and test datasets
```

```
#### Split non-preprocessed data
```

```
"""
```

```
X_train_npp, X_test_npp, y_train_npp, y_test_npp = train_test_split(X_npp,  
y_unenc,\
```

```
test_size=0.33, random_state=42)
```

```
print(X_train_npp.shape, y_train_npp.shape)
```

```
print(X_test_npp.shape, y_test_npp.shape)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y_unenc,\
```

```
test_size=0.33, random_state=42)
```

```
print(X_train.shape, y_train.shape)
```

```
print(X_test.shape, y_test.shape)
```

```
X_test.head()
```

```
"""#### Trying to send data to the endpoint will return predictions with  
probabilities
```

```
### 11. Scale our data
```

```
"""
```

```
# Use StandardScaler
```

```
scaler = preprocessing.StandardScaler().fit(X_train, y_train)
```

```
X_train_scaled = scaler.transform(X_train)
```

```
pd.DataFrame(X_train_scaled, columns=X_train.columns).head()
```

```
pd.DataFrame(y_train).head()
```

```
"""### 12. Start building a classifier
```

```
#### Support Vector Macines on non-preprocessed data
```

```
"""
```

```
from sklearn.svm import SVC
```

```
# Run classifier
```

```
clf_svc_npp = svm.SVC(random_state=42)
```

```
clf_svc_npp.fit(X_train_npp, y_train_npp)
```

```
#### Support Vector Machines on preprocessed data"""
```

```
from sklearn.linear_model import LogisticRegression
```

```
# Run classifier
```

```
clf_svc = svm.SVC(random_state=42)
```

```
clf_svc.fit(X_train_scaled, y_train)
```

```
#### Logestic Regression on preprocessed data"""
```

```
from sklearn.linear_model import LogisticRegression
```

```
clf_lr = LogisticRegression()
```

```
model = clf_lr.fit(X_train_scaled, y_train)
```

```
model
```

```
#### Multilayer Perceptron (Neural Network) on preprocessed data"""
```

```
from sklearn.neural_network import MLPClassifier
```

```
clf_mlp = MLPClassifier(verbose=0)
clf_mlp.fit(X_train_scaled, y_train)
```

Note: MLP as a NN, can use data without the feature engineering step, as the NN will handle that automatically

```
"""### 13. Evaluate our model"""
```

```
# Use the scaler fit on trained data to scale our test data
X_test_scaled = scaler.transform(X_test)
pd.DataFrame(X_test_scaled, columns=X_train.columns).head()
```

```
"""#### Evaluate SVC on non-preprocessed data"""
```

```
# Predict confidence scores for data
y_score_svc_npp = clf_svc_npp.decision_function(X_test_npp)
pd.DataFrame(y_score_svc_npp)
```

```
# Get accuracy score
from sklearn.metrics import accuracy_score
y_pred_svc_npp = clf_svc_npp.predict(X_test_npp)
acc_svc_npp = accuracy_score(y_test_npp, y_pred_svc_npp)
print(acc_svc_npp)
```

```
# Get Precision vs. Recall score
from sklearn.metrics import average_precision_score
average_precision_svc_npp = average_precision_score(y_test_npp,
y_score_svc_npp)
```

```
print('Average precision-recall score: {0:0.2f}'.format(
    average_precision_svc_npp))
```

```
"""#### Evaluate SVC on preprocessed data"""
```

```
# Get model confidence of predictions
```

```
y_score_svc = clf_svc.decision_function(X_test_scaled)
```

```
y_score_svc
```

```
# Get accuracy score
```

```
y_pred_svc = clf_svc.predict(X_test_scaled)
```

```
acc_svc = accuracy_score(y_test, y_pred_svc)
```

```
print(acc_svc)
```

```
# Get Precision vs. Recall score
```

```
average_precision_svc = average_precision_score(y_test, y_score_svc)
```

```
print('Average precision-recall score: {0:0.2f}'.format(  
    average_precision_svc))
```

```
"""#### Evaluate Logistic Regression on preprocessed data"""
```

```
y_score_lr = clf_lr.decision_function(X_test_scaled)
```

```
y_score_lr
```

```
y_pred_lr = clf_lr.predict(X_test_scaled)
```

```
acc_lr = accuracy_score(y_test, y_pred_lr)
```

```
print(acc_lr)
```

```
average_precision_lr = average_precision_score(y_test, y_score_lr)
```

```
print('Average precision-recall score: {0:0.2f}'.format(  
    average_precision_lr))
```

```
"""#### Evaluate MLP on preprocessed data"""
```

```
y_score_mlp = clf_mlp.predict_proba(X_test_scaled)[: , 1]  
y_score_mlp
```

```
y_pred_mlp = clf_mlp.predict(X_test_scaled)  
acc_mlp = accuracy_score(y_test, y_pred_mlp)  
print(acc_mlp)
```

```
average_precision_mlp = average_precision_score(y_test, y_score_mlp)
```

```
print('Average precision-recall score: {0:0.2f}'.format(  
    average_precision_mlp))
```

```
"""### 14. ROC Curve and models comparisons"""
```

```
# Plot SVC ROC Curve
```

```
plt.figure(0, figsize=(20,15)).clf()
```

```
fpr_svc_npp, tpr_svc_npp, thresh_svc_npp = metrics.roc_curve(y_test_npp,  
y_score_svc_npp)
```

```
auc_svc_npp = metrics.roc_auc_score(y_test_npp, y_score_svc_npp)  
plt.plot(fpr_svc_npp, tpr_svc_npp, label="SVC Non-Processed, auc=" +  
str(auc_svc_npp))
```

```
fpr_svc, tpr_svc, thresh_svc = metrics.roc_curve(y_test, y_score_svc)  
auc_svc = metrics.roc_auc_score(y_test, y_score_svc)  
plt.plot(fpr_svc, tpr_svc, label="SVC Processed, auc=" + str(auc_svc))
```

```
fpr_mlp, tpr_mlp, thresh_mlp = metrics.roc_curve(y_test, y_score_mlp)  
auc_mlp = metrics.roc_auc_score(y_test, y_score_mlp)  
plt.plot(fpr_mlp, tpr_mlp, label="MLP, auc=" + str(auc_mlp))
```



```
fpr_lr, tpr_lr, thresh_lr = metrics.roc_curve(y_test, y_score_lr)
auc_lr = metrics.roc_auc_score(y_test, y_score_lr)
plt.plot(fpr_lr, tpr_lr, label="Logistic Regression, auc=" + str(auc_lr))

plt.legend(loc=0)
```

```
filename = 'clf_svc.pkl'
pickle.dump(clf_svc, open(filename, 'wb'))
#!mkdir C:\Users\Palani\Downloads\model
!cp clf_svc.pkl C:\Users\Palani\Downloads
!tar -zcvf clf_svc.tar.gz clf_svc.pkl
```

```
from ibm_watson_machine_learning import APIClient
```

```
wml_credentials = {
    "url": "https://eu-gb.ml.cloud.ibm.com",
    "apikey": "***"
}
```

```
client = APIClient(wml_credentials)
```

```
metadata = {
    client.spaces.ConfigurationMetaNames.NAME: "Telco Churn DS",
    client.spaces.ConfigurationMetaNames.DESRIPTION: "To predict
customers who exit the company",
    client.spaces.ConfigurationMetaNames.STORAGE: {
        "type": "bmcos_object_storage",
        "resource_crn": "***"
    },
    client.spaces.ConfigurationMetaNames.COMPUTE: {
        "name": "WatsonMachineLearning",
```

```
        "crn": "***"
    },
}
```

```
space_details = client.spaces.store(meta_props=metadata)
```

```
space_details
```

```
space_id = space_details["metadata"]["id"]
space_id
```

```
#space_id = "***"
```

```
client.set.default_space(space_id)
```

```
client.software_specifications.list()
```

```
import sklearn
sklearn.__version__
```

```
spec_id =
client.software_specifications.get_id_by_name("scikit-learn_0.20-py3.6")
```

```
#spec_id = "***"
```

```
model_details = client.repository.store_model(model=clf_svc,meta_props={
    client.repository.ModelMetaNames.NAME:"Churn Prediction",
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:spec_id,
    client.repository.ModelMetaNames.TYPE:"scikit-learn_0.20"
})
```

```
model_id = model_details["metadata"]["id"]
```

model_id

```
#model_id = "****"
```

```
deployment_metadata = {  
    client.deployments.ConfigurationMetaNames.NAME:"Churn Prediction  
Deployment",  
    client.deployments.ConfigurationMetaNames.ONLINE:{  
}
```

```
deployment_details = client.deployments.create(artifact_uid=model_id,  
meta_props=deployment_metadata)
```

```
deployment_id = deployment_details["metadata"]["id"]
```

```
col = X.columns  
col = list(col)  
col
```

```
score_list = ['589','39','6','163520.37','3','1','0','75238.55','0','1','0','1','0']
```

```
payload = {  
    client.deployments.ScoringMetaNames.INPUT_DATA:({  
        "fields":col,  
        "values":[score_list],  
    })  
}
```

```
deployment_details =  
client.deployments.score(deployment_id=deployment_id,  
meta_props=payload)
```

```
deployment_details
```