**6006CEM MACHINE LEARNING REPORT**

**Coventry GitHub Repository URL** or **Coventry OneDrive URL** (mandatory):

https://github.coventry.ac.uk/liy332/11384841-YL-s1

Contents

[6006CEM MACHINE LEARNING REPORT 1](#_Toc7814)

[1. Introduction 2](#_Toc29652)

[1.1 Background and code structure 2](#_Toc27225)

[1.2 Comparison with existing work 2](#_Toc14764)

[1. More ways to represent results 2](#_Toc12418)

[2. Project implementation 4](#_Toc25925)

[2.1data analysis process 4](#_Toc25628)

[2.1.1 Import data and the first five rows of the dataset 4](#_Toc5480)

[2.1.2 Check data for missing values 5](#_Toc10368)

[2.1.3 Find outliers in the data set 9](#_Toc10483)

[2.1.4 Data visualization 11](#_Toc11861)

[2.1.5 Data Preprocessing 18](#_Toc11820)

[2.1.6 Building the algorithm model 21](#_Toc21021)

[2.1.7 Model Turning 27](#_Toc16795)

[2.1.8 model evaluation 29](#_Toc20515)

[2.2 Conclusion 32](#_Toc178)

|  |
| --- |
| **Academic Report** |

# Introduction

## 1.1 Background and code structure

This report will introduce the use of machine learning to solve the real-world classification problem of telecommunication customer churn. Predicting customer churn is a very important issue for enterprises in the display world. Through data analysis, we can master the factors related to customer churn rate, locate potential customers who are lost, and then maintain customers in a targeted manner.

Accurately grasp the important factors related to customer churn and find solutions to maximize the company's benefits.

The report is to use machine learning to conduct research on the data set, and the specific steps include the following steps

1. Analyze each data class in the data set to understand the value and distribution of each data item in the data set.

2. Use boxplot, heatmat, histplot and other graphics to realize data visualization --- Compare related data in a more three-dimensional manner, and find factors related to customer churn rate.

3. Use encoder/MinMaxScakler/Smote and other methods to preprocess the data to be analyzed to prepare for model building

4. Use different algorithms to build learning models --- Logistic regression/ Gradient Boosting Classifier/ KNN

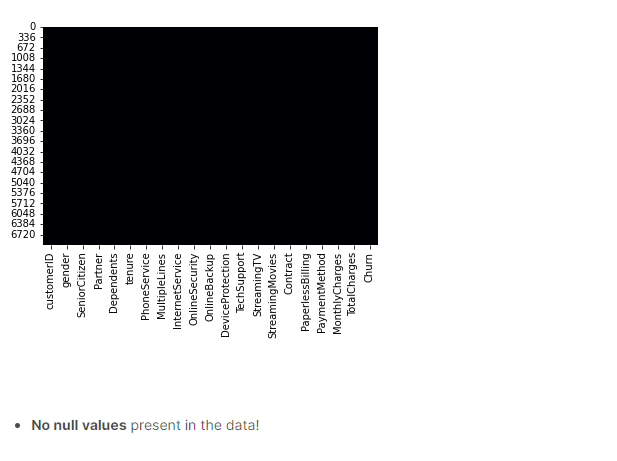
1. Improve the performance of existing models and evaluate them

## 1.2 Comparison with existing work

By comparing it with the work of TANMAY DESHPANDE.(Deshpande, T. (2022)), I found that basically the process is close to the same. But there are still some differences

1. More ways to represent results

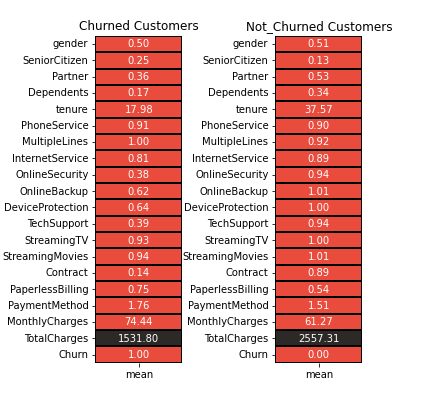
By contrast, in my task, I output most of the data conclusive information in the form of text, and do not use much data visualization. For example, in his task, he chose to use a Heatmap -- which looks for null in the dataset. In this way, people can quickly explore the data through the icon and intuitively notice the information they want to express.



-- Screenshot for his work

2. Explore more relationships between features in data visualization

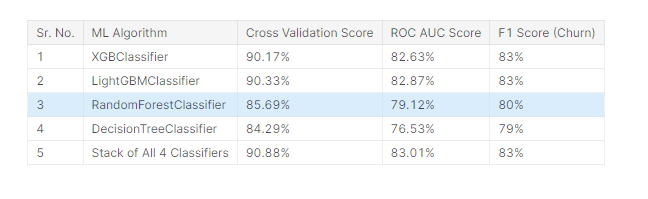
Through comparison, I found that although my data visualization covered most of the graphic types, I did not spend more experience exploring the internal relationship between different features in the data set. For example, in his task, he charts and compares the impression factors of yes and no in churn with other different data in the database. But in my data visualization, I use more pie charts to show the proportion of options between each feature.



-- Screenshot for his work

3. More evaluation methods and algorithm models are used

For his task, he evaluated the model using corss validation score and ROC\_AUG scorss and F1 score, whereas I evaluated it using the obfuscation matrix and F1 score. In his algorithm summary table, he found more algorithm models (and more complex ones). However, in the final F1 score comparison, I found that the model score was almost the same as his -- but he did not make the model for turning, so maybe my data preprocessing was not very good.

 -- Screenshot for his work

# Project implementation

## 2.1data analysis process

### Import data and the first five rows of the dataset

First, we will import the dataset WA\_Fn-UseC\_-Telco-Customer-Churn.csv file and check the first five rows of data in the dataset to ensure the correctness of the imported dataset. Next we found that in Figure 2, we can clearly find another 7043 rows and 21 columns in the dataset. And the data types include object, int64 and float64.

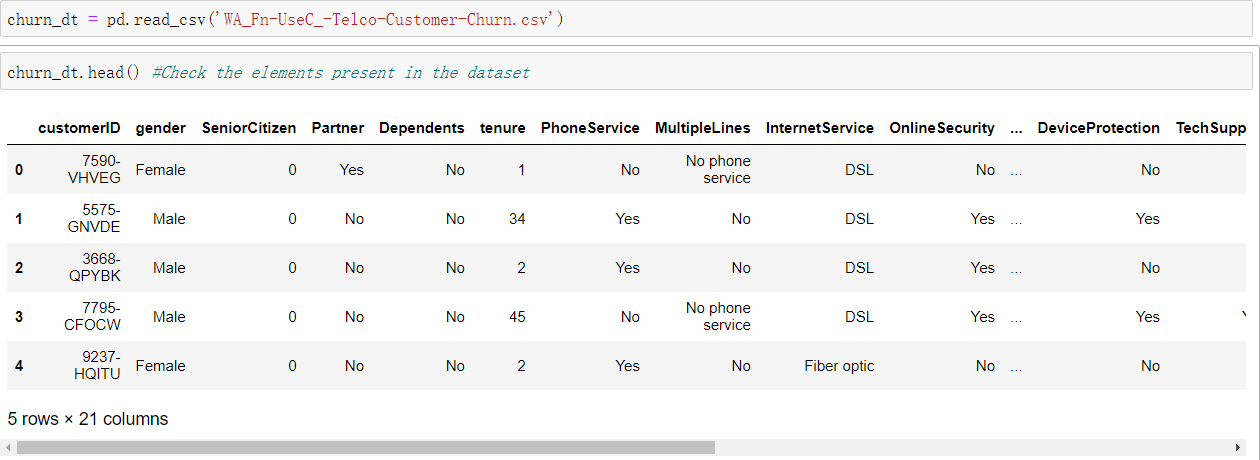


Figure 1 -- Import data and the first five rows of the dataset

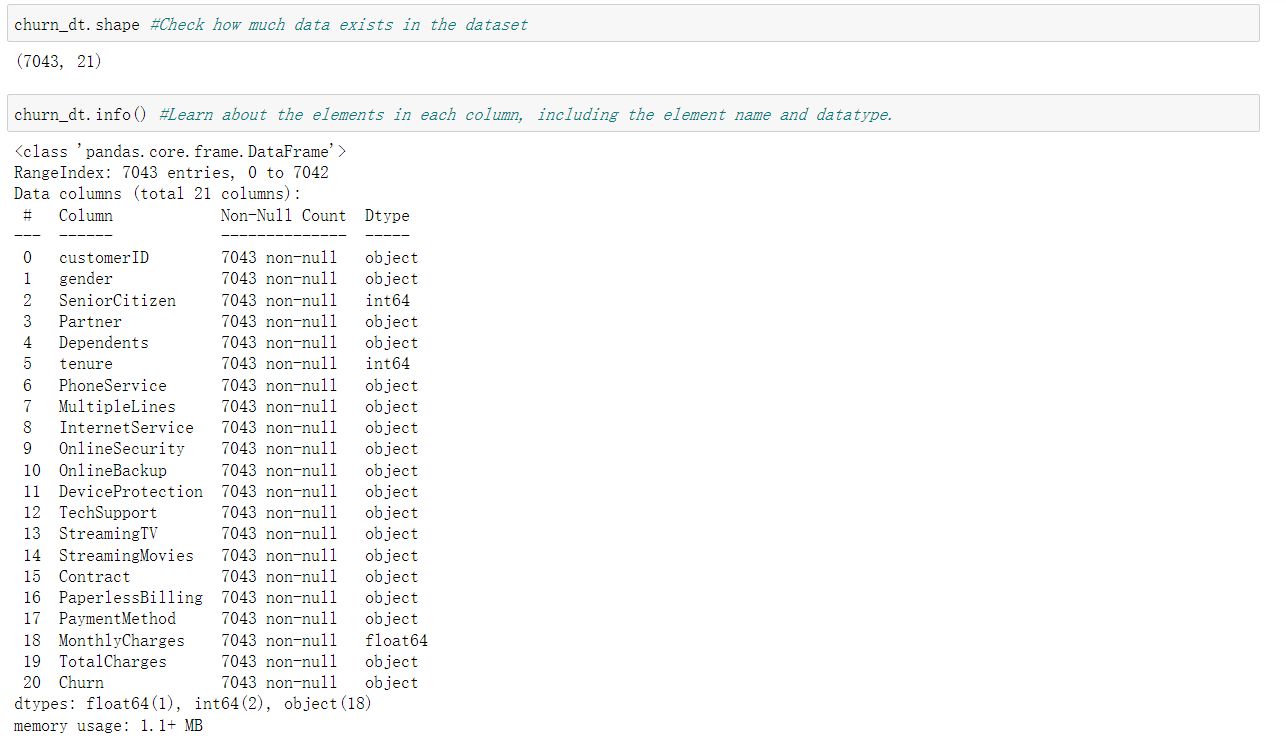
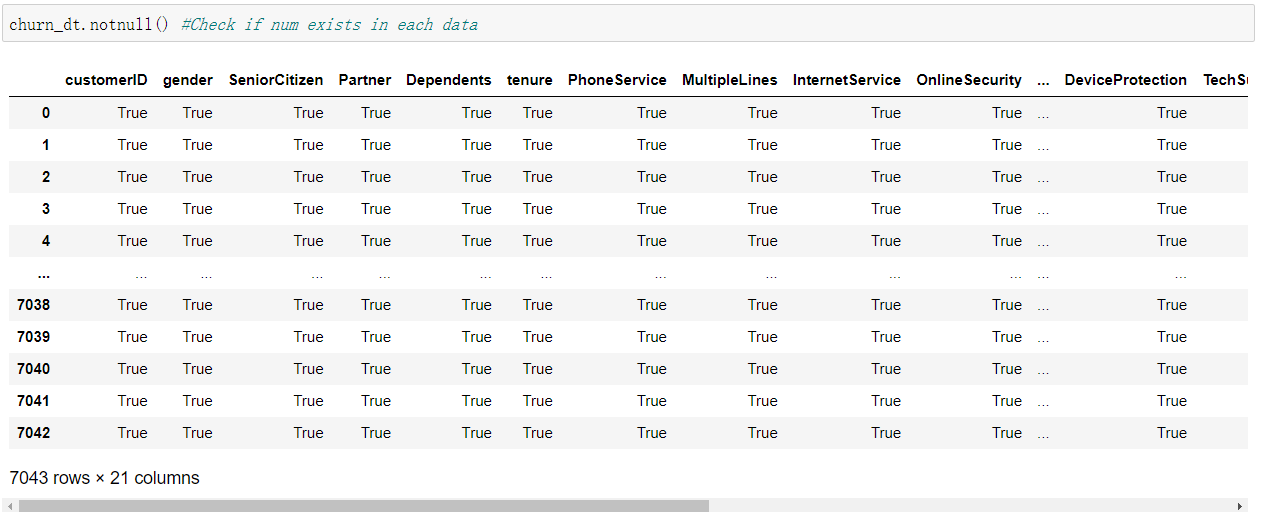


Figure 2 -- The number and type of columns in the data

### Check data for missing values

After the data check is completed, check the data. First, we check for the null value of the data. The first is to check if the table has a null value by the notnull() built-in function. According to the output, there are no null values in the visible rows and columns. Then use the isnull() function to check each element in the dataset, and the results all show that there is no null value.



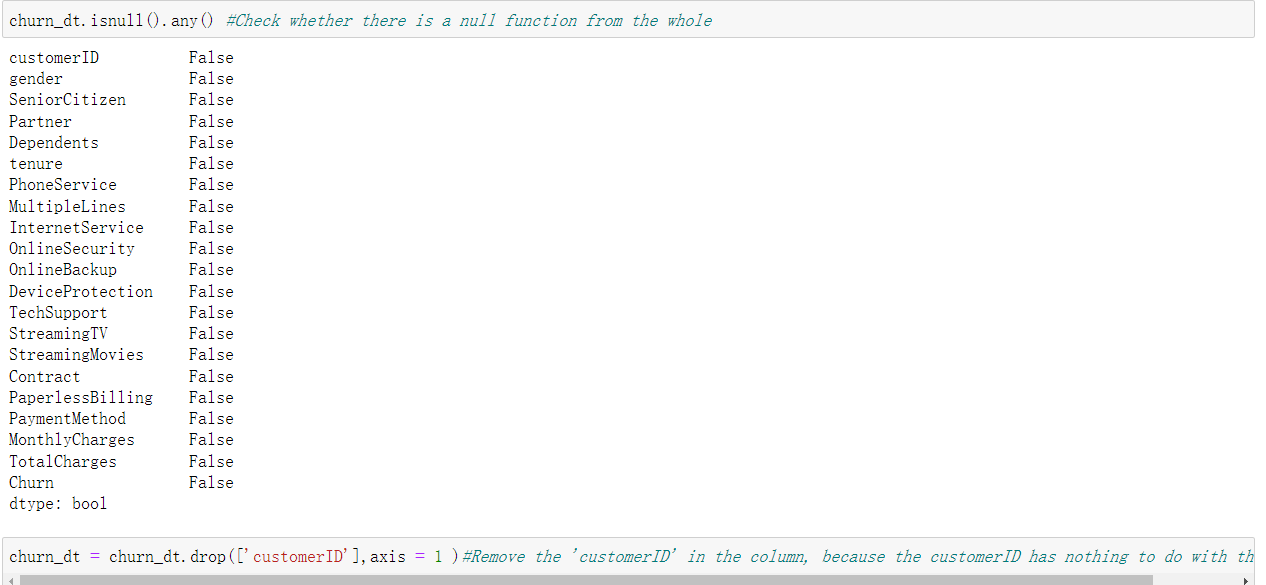


Figure 3 -- check data have null value

In Figure 1, I found that the data type of the TotalCharges element is object, but after checking, the data here is all presented in a numeric type. In order to avoid indistinguishable float values, the to\_numeric function is used to convert the float type. Checked again and found 11 null values.

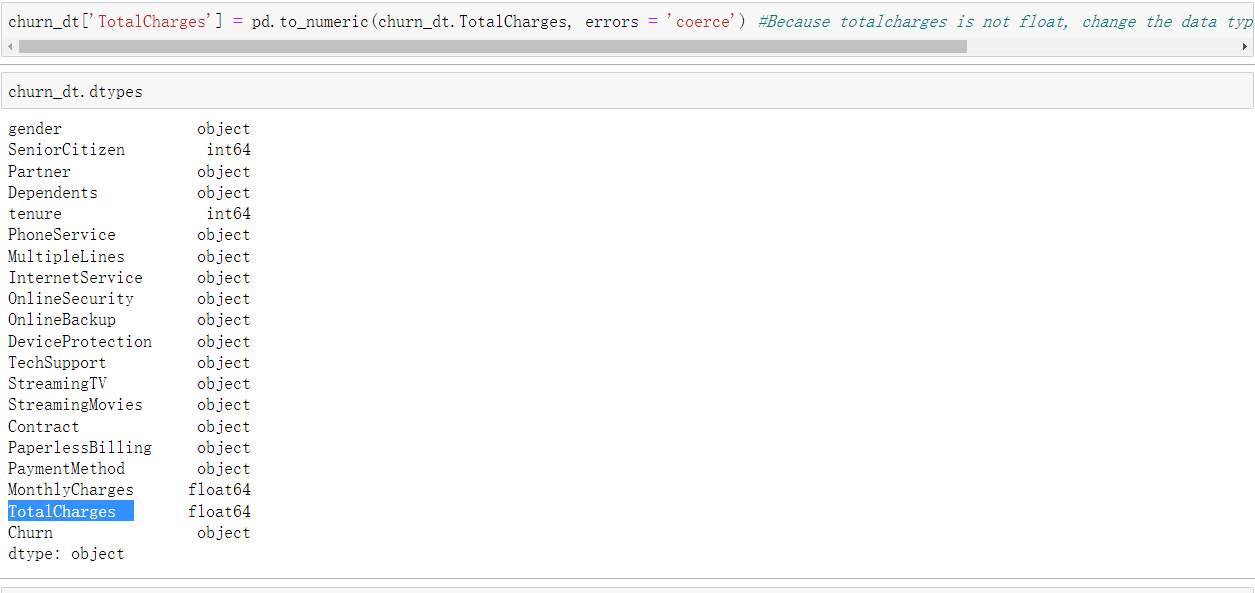
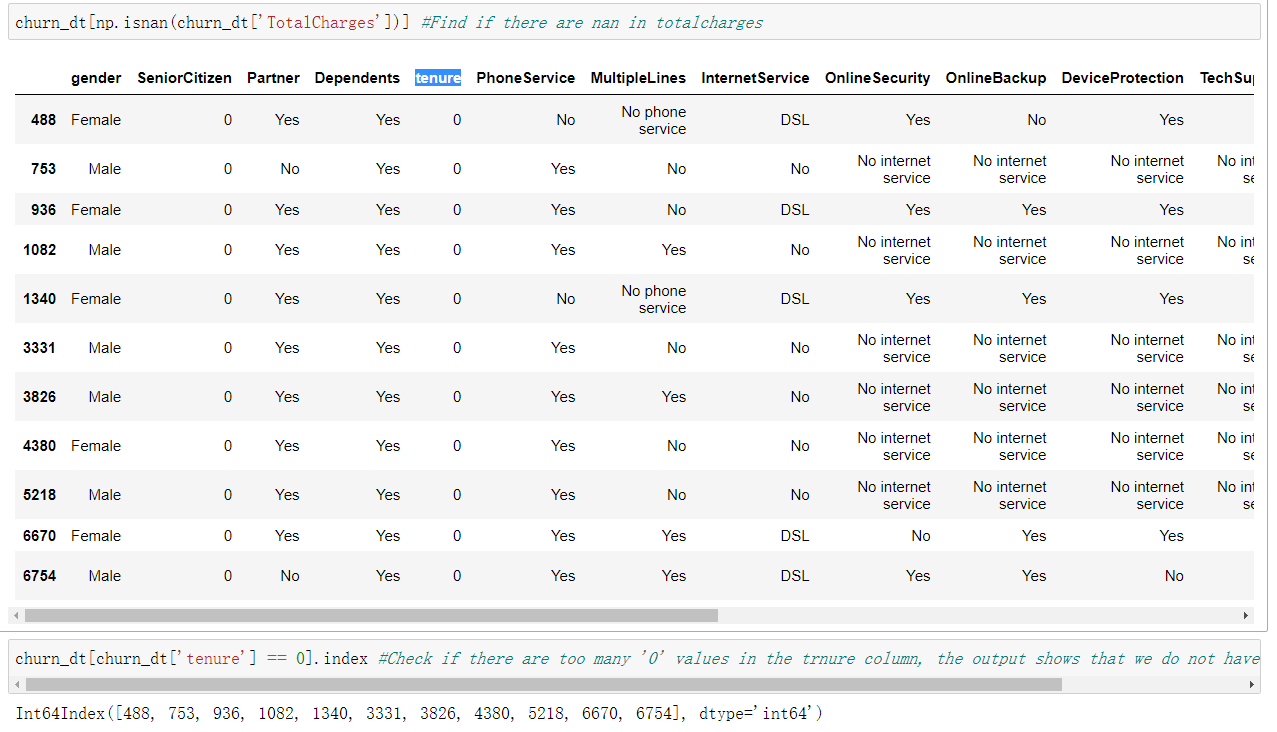




Figure 4 -- find the missing value and change data type

Use the isnan function to view the specific data of null, and find that there are many missing values in 'tenure'. Considering that there are only eleven missing values, removing them does not affect the analysis of the data (Figure 5). Because TotelCharge contains Nan, but in order to ensure the smoothness of the next analysis, I added all the average values of this element to ensure the accuracy of the analysis (Figure 6).



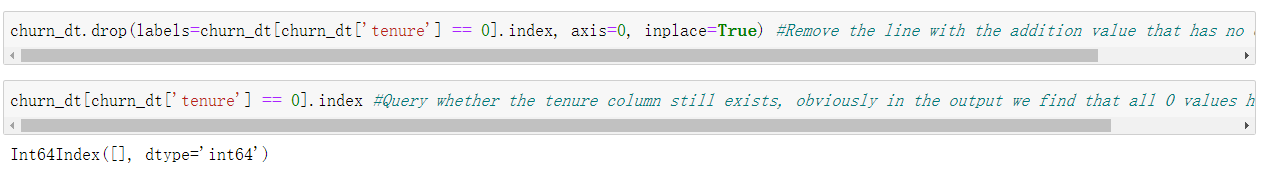
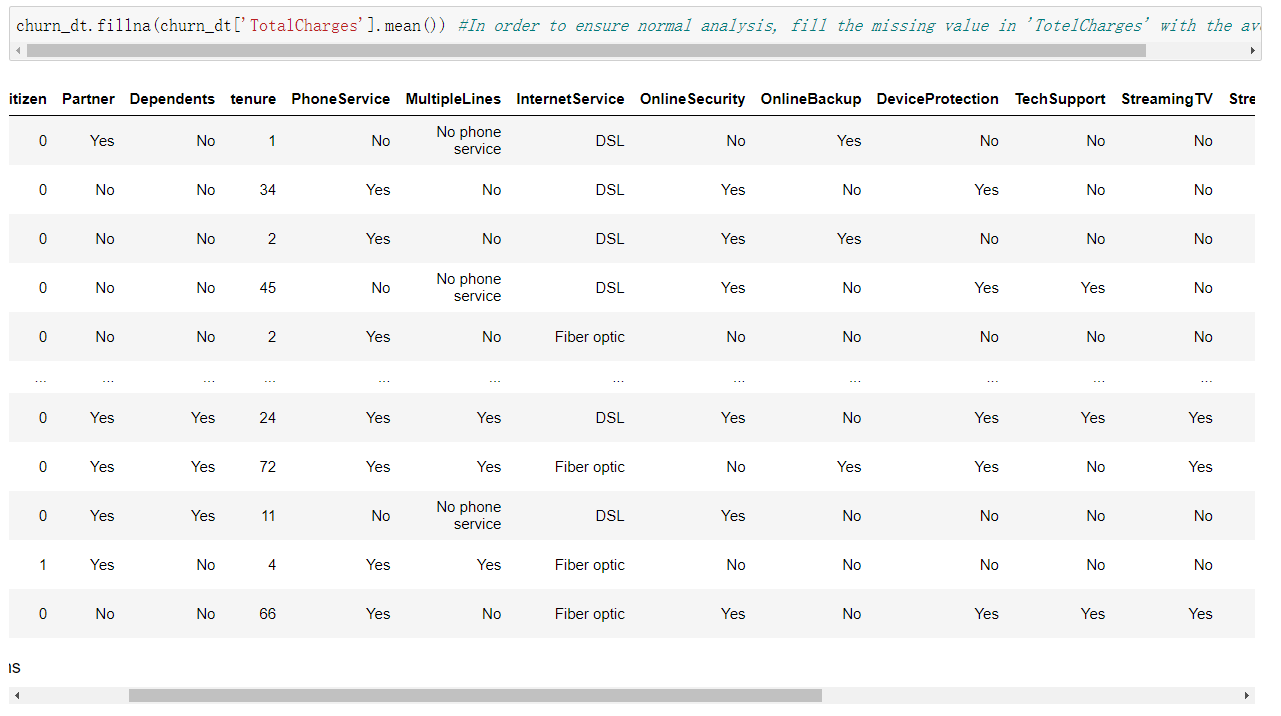


Figure 5 -- Look for and remove missing values from 'tenure'



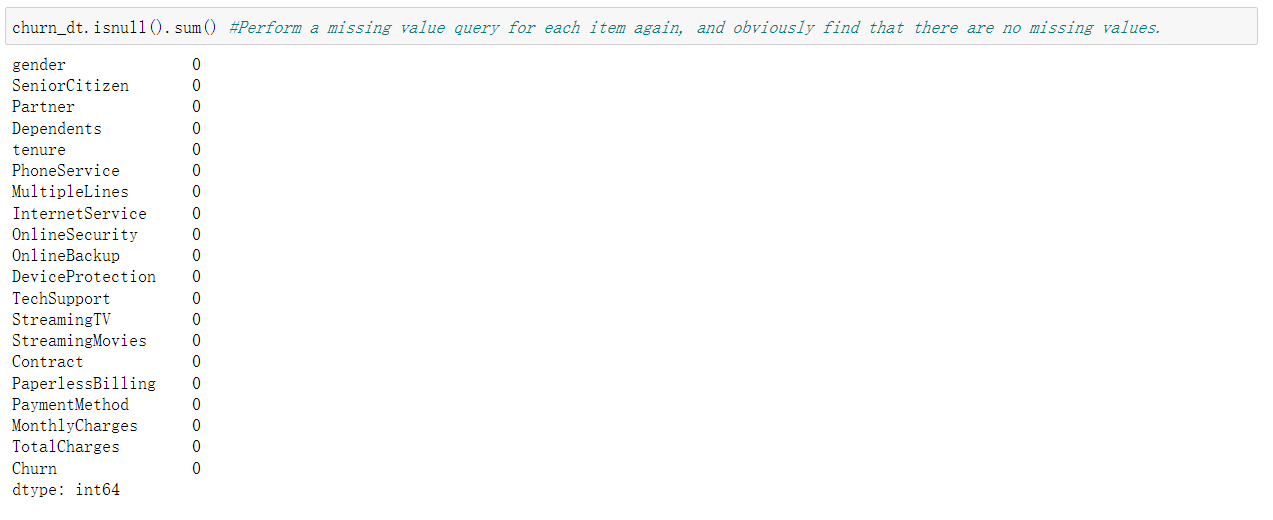


Figure 6 -- Populate the Nan with an average value

### Find outliers in the data set

Next, in order to ensure the smooth analysis of the data, we need to find outliers in the data.

Outliers usually refer to the presence of too large or too small values in the data of a certain element in the data set. Outliers may be due to experimental error; the latter are sometimes excluded from the dataset. Outliers can cause severe output bias in the data analysis part of machine learning.

Next, in order to ensure the smooth analysis of the data, we need to find outliers in the data.

Outliers usually refer to the presence of too large or too small values in the data of a certain element in the data set. Outliers may be due to experimental error; the latter are sometimes excluded from the dataset. Outliers can cause severe output bias in the data analysis part of machine learning(Outlier. (n.d.)).

Use the skew() function to help find outliers. Skew refers to the degree of distortion in the normal distribution. If skewness value greater than 1 or less than -1 indicates a highly skewed distribution, between 0.5 and 1 or -0.5 and -1 is moderately skewed(Oracle® Crystal Ball Reference and Examples Guide. (n.d.)).

First, I used box plot(Venmani, a d. (2020)) to display the values of ''MonthlyCharges', 'tenure', 'TotalCharges' in a graphical form, and I can clearly see the distribution of the data. It is clearly seen in Figure 7 that there are outliers in tenure and TotalCharges.

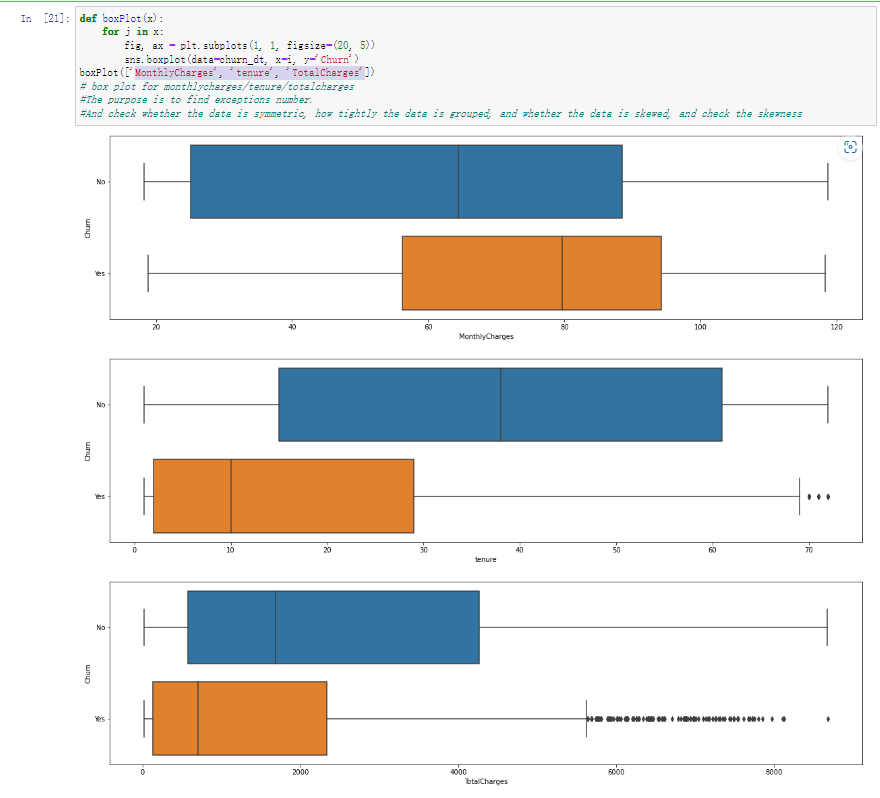


Figure 7 -- Use boxplot to find outliers

According to the figure, I chose to drop the value greater than 7800 in totalcharges (the real outlier), and used the skew function to check, and found that the skewness value(.mcneese, B. (2008)) of totalcharges only dropped by 0.01, so we do not consider it an outlier. So this will not be in the this machine learning program. SeniorCitizen is not outliers because it is represented by 0 and 1.

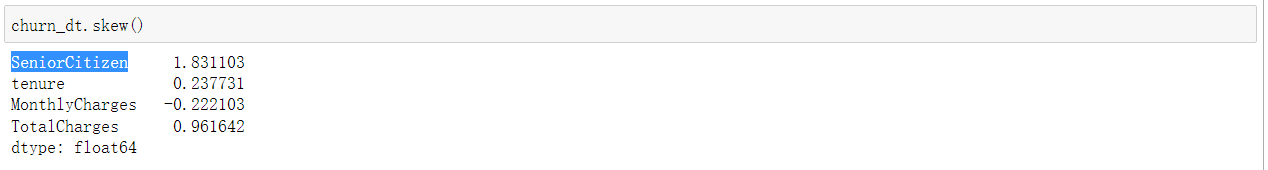


Figure 8 -- Look for outliers before

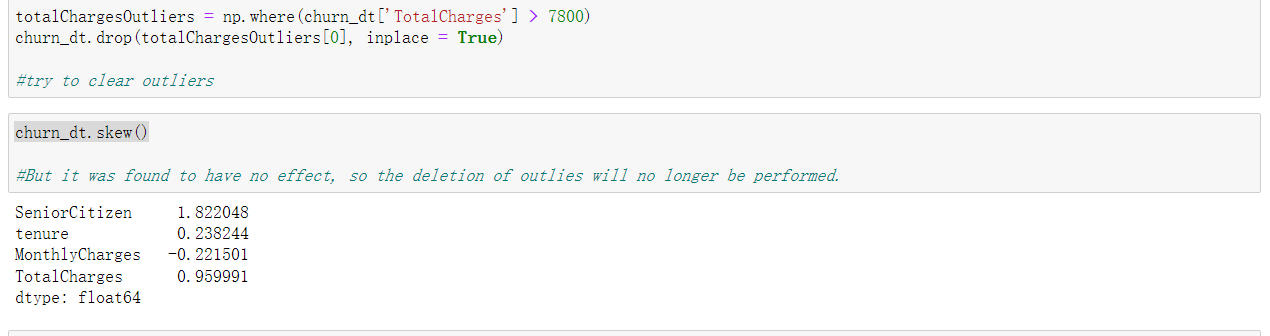


Figure 9 -- Look for outliers after

### Data visualization

To make the data more vivid, we will use views to visualise some of the highlights of the data. As not all data in a dataset is evenly distributed, data visualisation gives a clearer picture of the distribution and partial balance of values in different data.

First, we present the relationship between 'churn' and 'gender' in the dataset using a histogram. In the code, we use the function histplot( from the seaborn library to show the univariate extent of the data



Figure 10 -- Use histogram to show ‘gender’ and ‘churn’

The data in Figure 9 demonstrates that with about equal numbers of males and females in this data, the proportion of churns is uneven, so churns are less correlated with gender. And we further validated our idea by plotting the counts with the seaborn library only countplot (Seaborn.)(Figure 11).

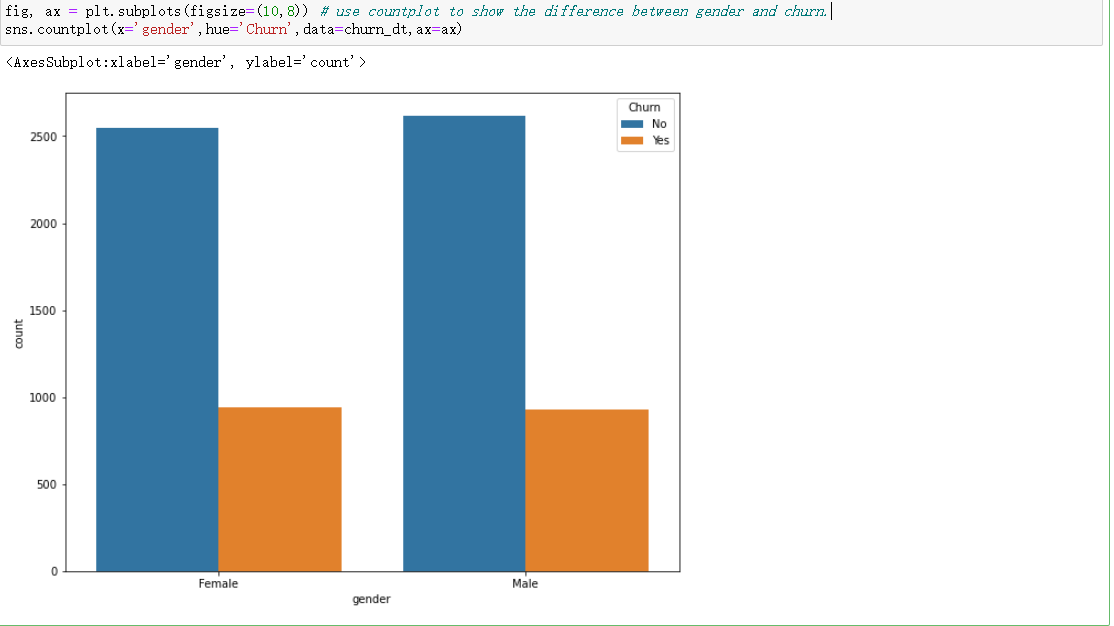
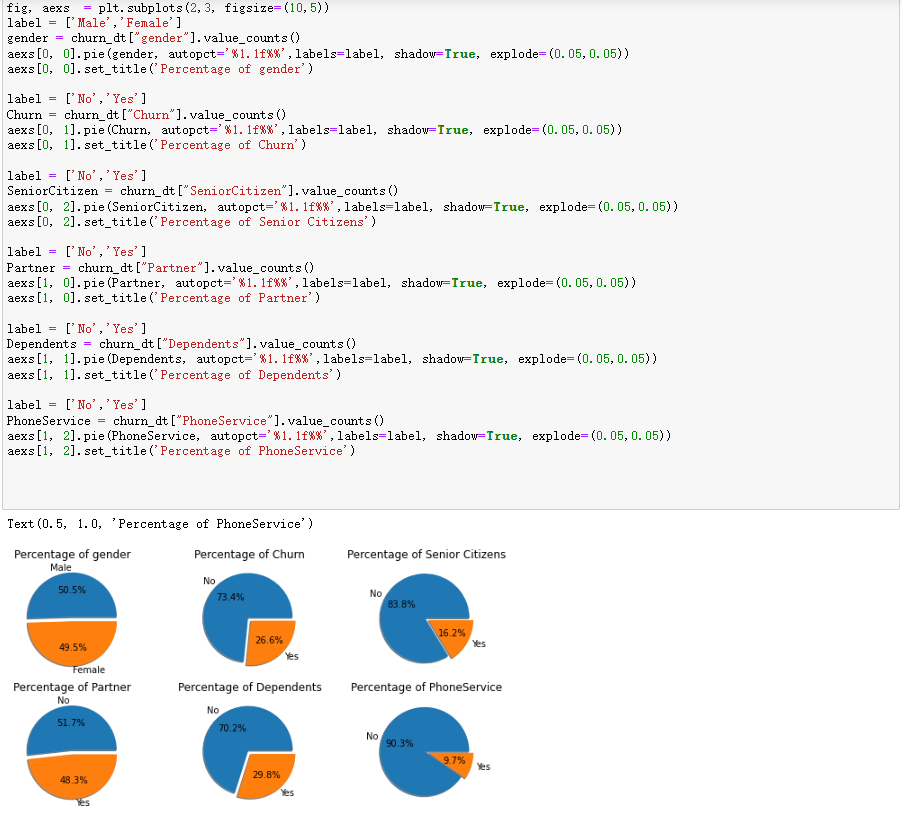


Figure11 -- countplot show relation between gender and churn

Next, we use a pie chart to implement the percentage distribution of data for the important parameters in the dataset. To achieve multiple plots, I am using subplots from the matplotlib library.



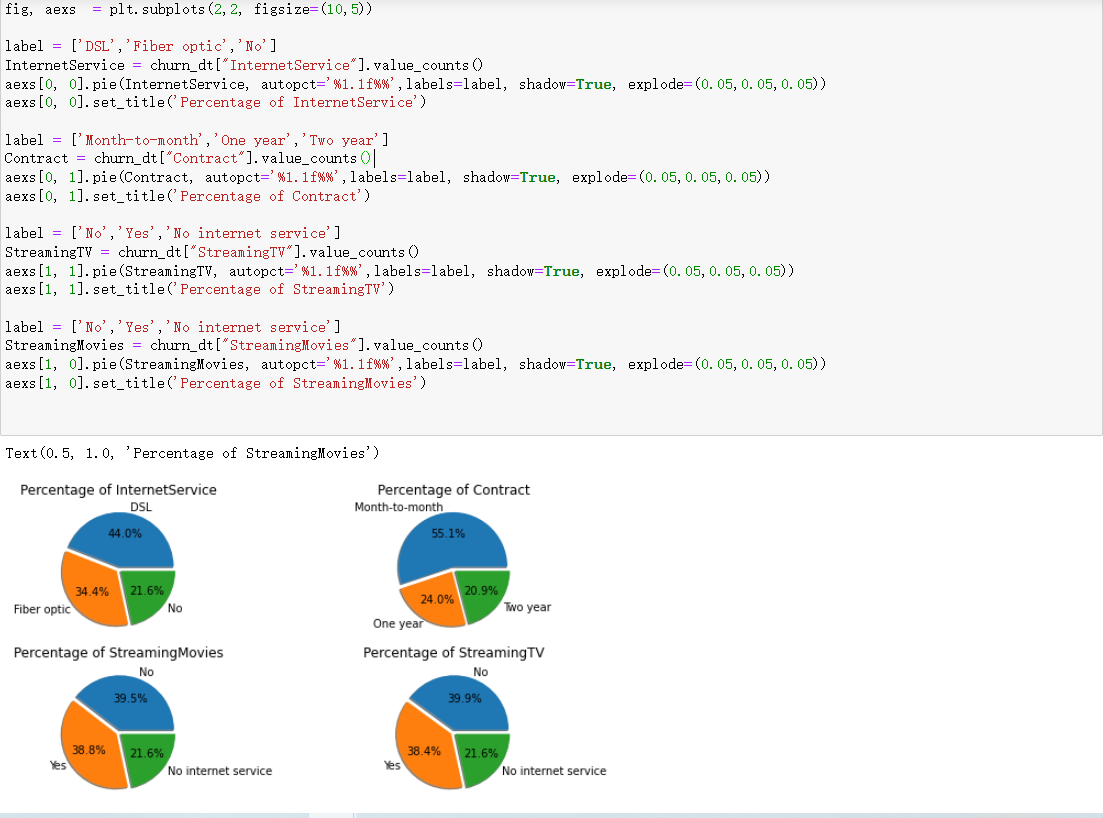


Figure 12 -- pie chart for feature detail

Figure Key illustration.

* In the graph it is clear that the vast majority of customers are non-elderly.
* Close to a quarter of users are lost.
* Close to 70% of customers have no dependents.
* The distribution of those using streaming movies and TV is close to the same percentage.
* More than half of customers use monthly payments.

In Figure 13, we plot the relationship between the churn and the customer contract using the histogram of the plotly.express module. From the graph, we can see that among the churn customers, most of them choose the monthly payment plan.

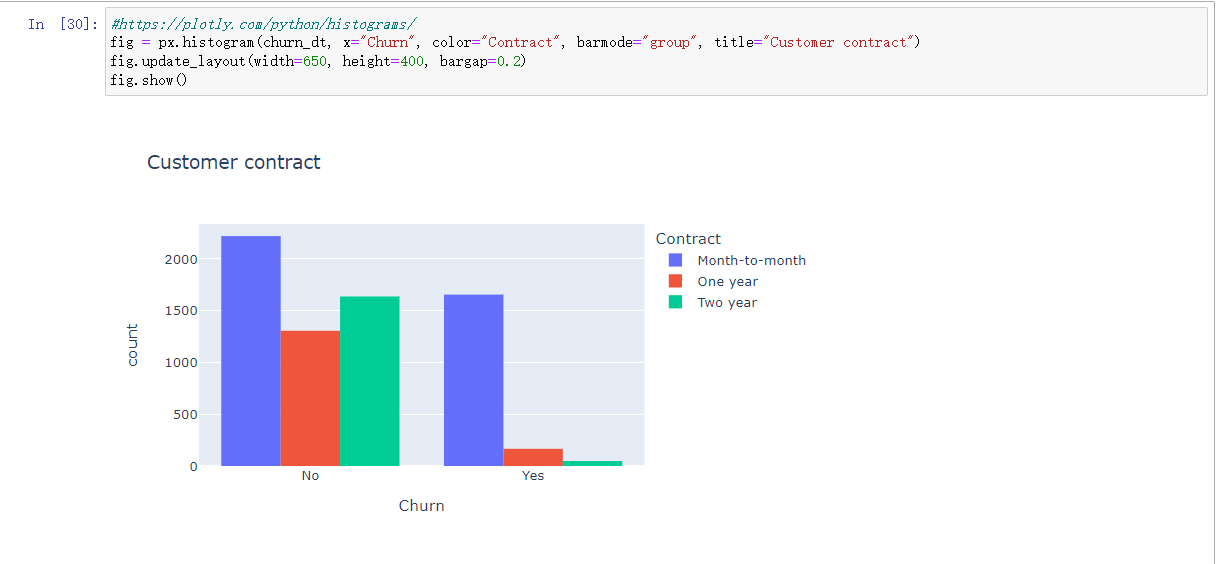


Figure 13 -- relation between churn and customer contract

In Figure 14, look for the relationship between user churn and Internet service and age. We use the plotly.graph\_objects module to compare the connection between internetservice and churn. Through the data, it is found that a large number of users choose fiber optic services, but the churn rate of these users is particularly high. This is evidence that there is a problem with this type of internet service business. Also, the churn rate of customers with DSL service will be lower than that of fiber optic service.



Figure14 -- relation between churn and international service

In Figure 15, we use scatterplot to find the relationship between churn and total and monthly fees. A scatterplot can display the relationship between two variables and visualize the strength of the relationship between the two variables. And found that the lost users are mostly pointed out that the cost is high and medium, and when the monthly cost is more than 70, the user is more willing to leave the plan.

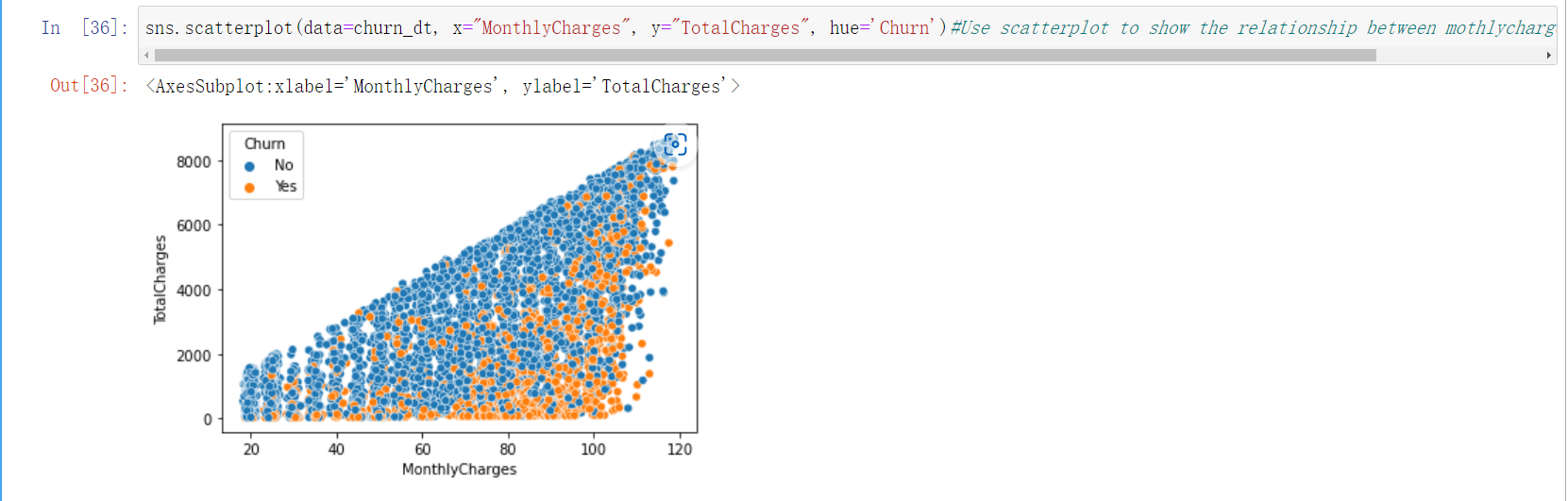


Figure15 - relation between churn and total/monthly charge

Next, we use heatmap to show the relationship between each feature. According to the value of the Pearson coefficient, the value approaches 0, and the smaller the relationship, the greater the opposite relationship. It is found that there is little relationship between senior citizen and tenure because the value approaches 0. There is a strong connection between tenure and totalcharge.

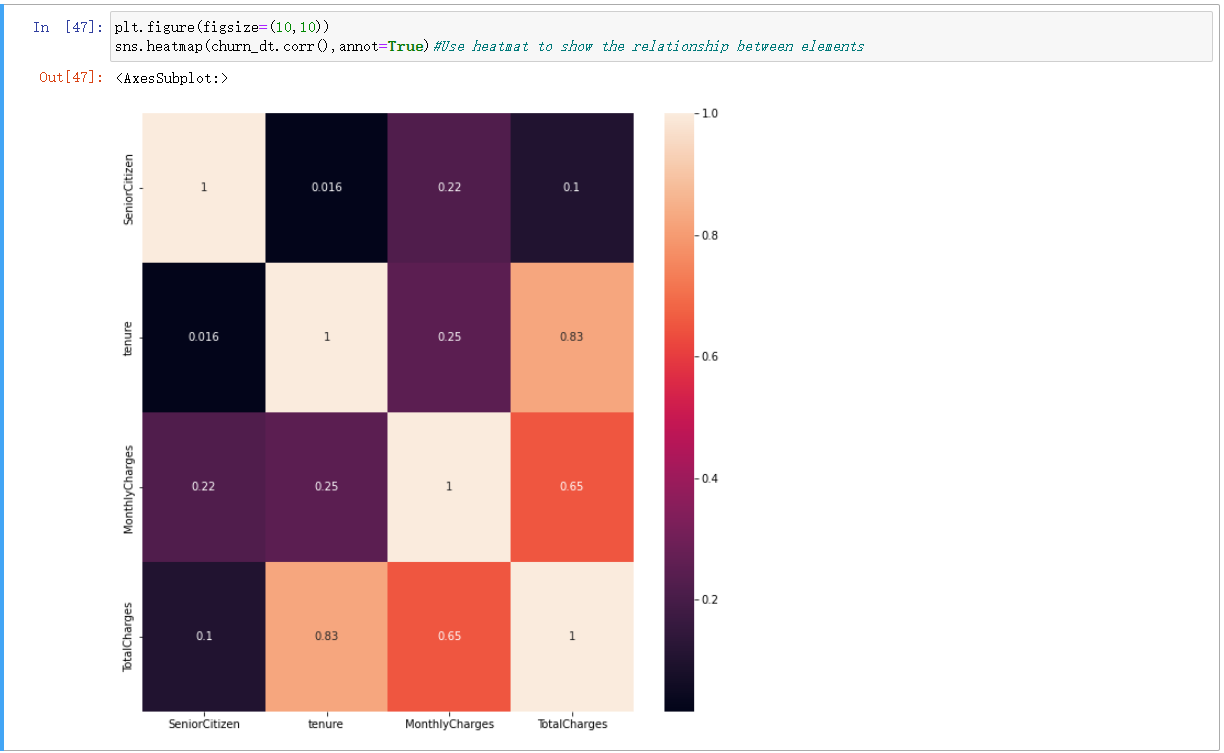


Figure 16 -- Relationships between different features

Finally, we use the kedplot module to plot the relationship between customer churn and monthly charge. It also proved once again that the monthly charges has exceeded 70, and the number of lost customers has increased again.

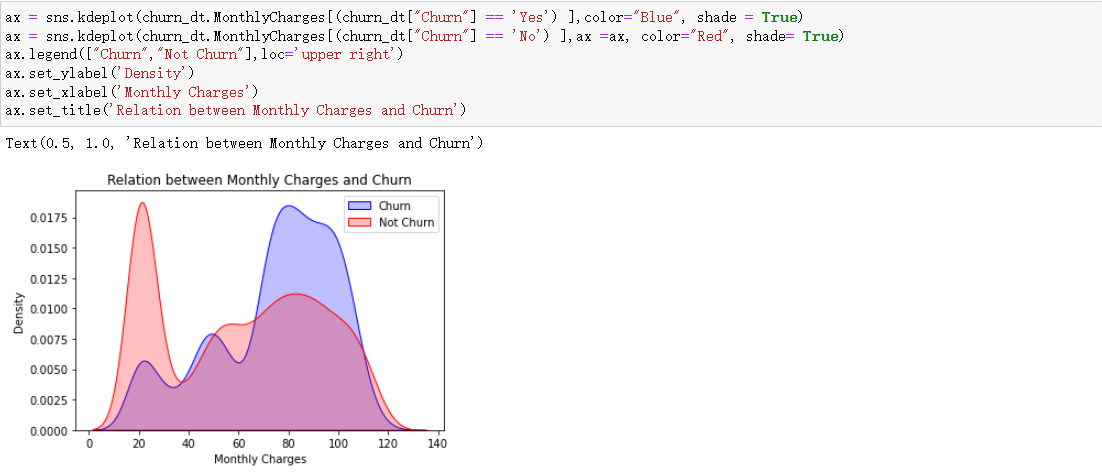


Figure 17 -- relation between monthly charges and churn

### Data Preprocessing

#### Data Transcoding

In classification datasets, there are many different types of data - including literal data (yes/no/etc). But in order for the computer to recognize it and the algorithmic model to recognize it, we need to transcode the data. In transcoding, I used a One-Hot encoder (since the same categorical data exists in different features) and I did the same transformation in integer encoding.

The advantage of the One-Hot encoder is that it solves the problem that the classifier is not good at handling attribute data, and its value is only 0,/1.

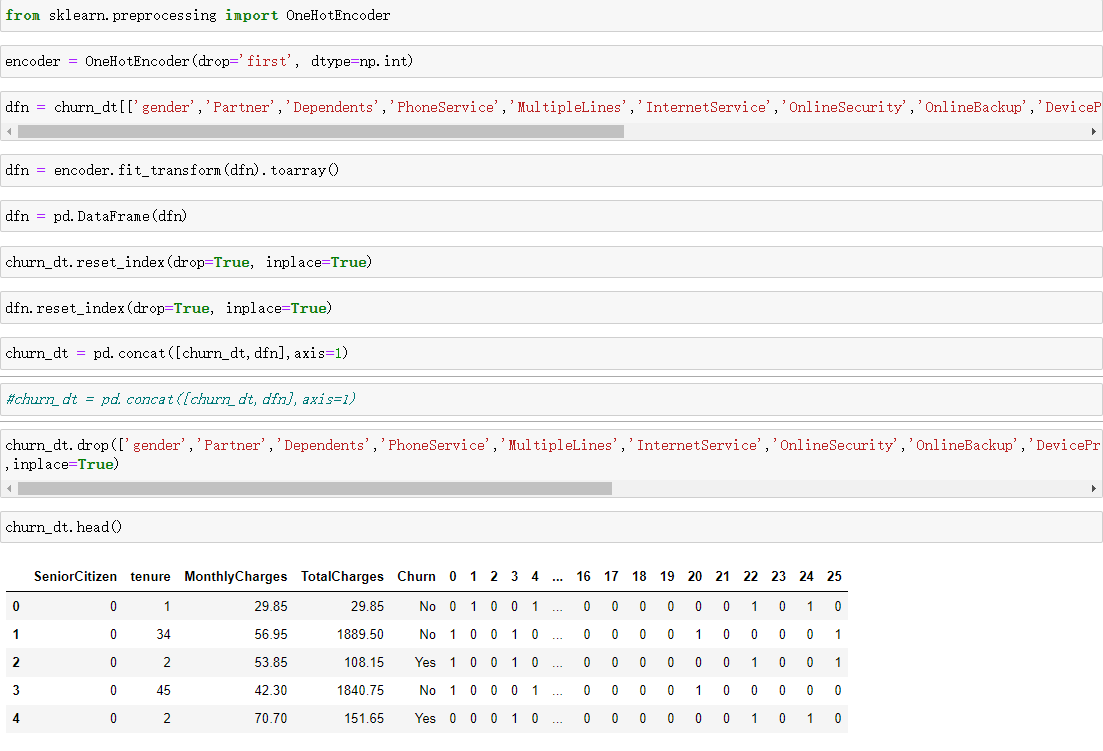


Figure 18 -- Ues One-Hot encoder to trancode the text type data

After conversion, we found that all non-numeric features have been converted to int32. But all our titles have been converted to digital form (which also does not convert the content).

Next, we perform a separate transformation on churn using the LabelEncoder module. The biggest difference from One-Hot Encoder is that it converts text into numerical values, (e.g. 0/1/2... in order). But based on churn, there is only No/Yes, so it will not affect.

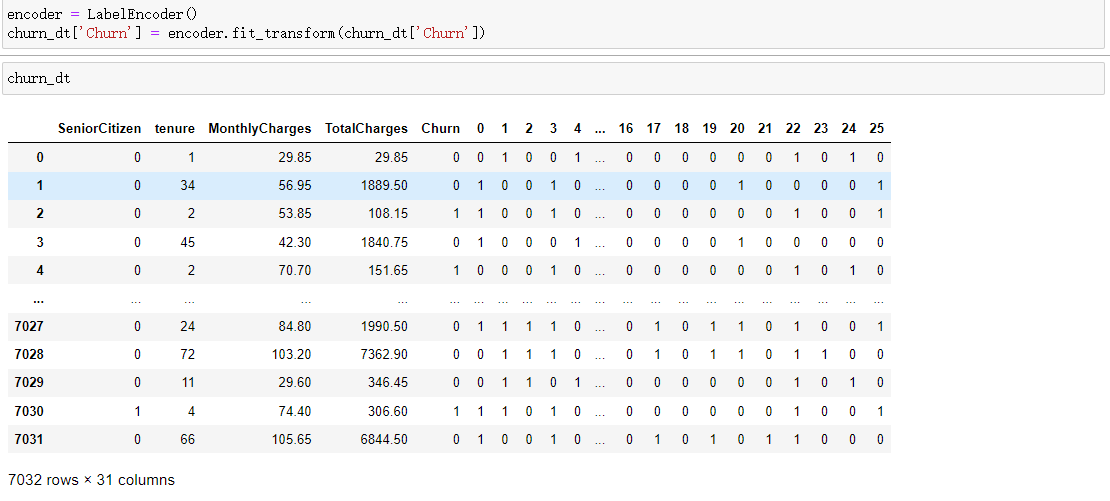


Figure 19 -- Ues Label Encoder to trancode ‘Churn’ feature

#### 2.1.5.2 Scaling Data-set feature

Next, we need to normalize the data. Data normalisation is a very important part of data pre-processing. In the dataset, there are some data with very large values. In machine learning, too large a scale between input variables can make many distance algorithm models more difficult (e.g. k-nearest neighbors algorithm), because increasing the scale of a feature is equivalent to increasing its weight in the distance calculation. This time I used MinMaxScaing(Brownlee , J. (2020)) for data normalisation, who could scaling the feature I had selected with amounts/terms to map the values to between [0, 1].

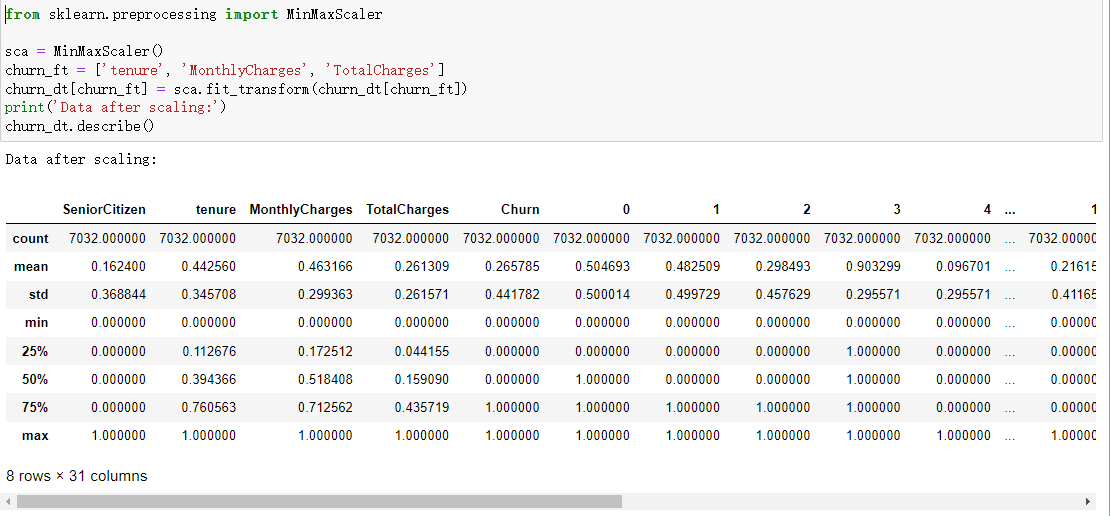


Figure 20 -- Use MinMaxScaler( ) to scaling the feature

#### 2.5.1.3 Balance Data

In categorical problem datasets, it can happen that the category-type dependent variable is heavily biased. In the data visualisation, I have found that the proportion of customers churned in the customer churn feature 'Churn' is only a quarter of the total. If the current data is used for prediction this may lead to biased conclusions. I will therefore use the imblearn library's SMOTE module, which analyses and simulates a small number of categories and adds new manually simulated samples to the data set to ensure that the proportions of the original data are not out of balance(Brownlee , J. (2021)).

First, we use the Counter module in the collections library, which will calculate the proportion before SMOTE. The data was then populated by SMOTE and then compared and found to be successful.

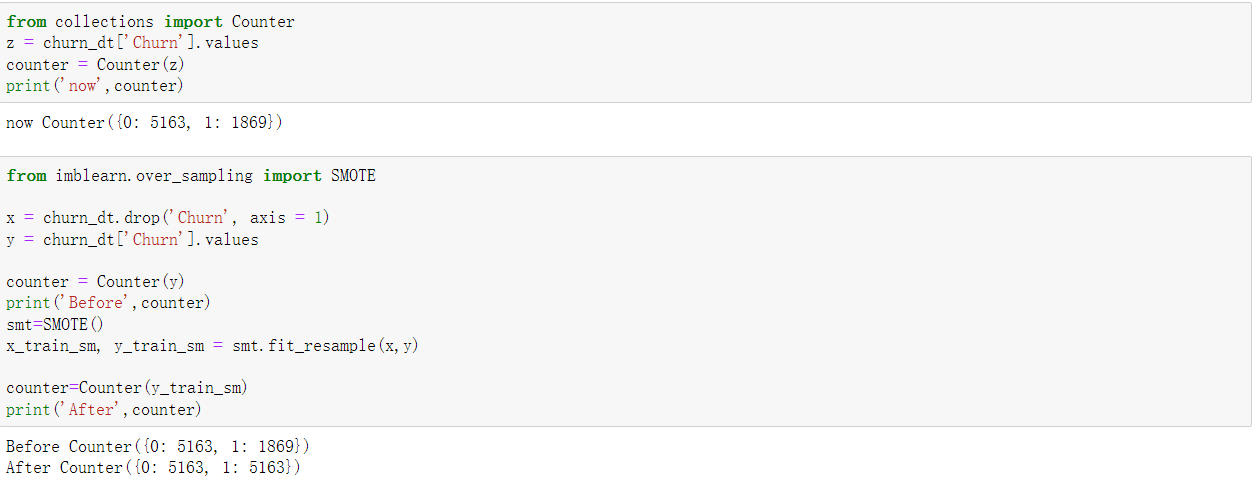


Figure 21 -- use SMOTE to balance data

#### 2.5.1.4 Create test/training data

Finally, before I start building the model, I divide the dataset into training data and test data. The training data is used for model building validation data to assist in the construction of the model and allow the machine to be trained. The test data is to test the construction of the model and to estimate the accuracy of the model.

I divide 30% for testing data and 70% for training data



Figure 22 -- create test/training data

### 2.1.6 Building the algorithm model

Once the preprocessing have been completed, the formal process of building the algorithmic model to implement the machine model begins.

In the re-model, I chose Logistic regression/Gradient Boosting Classifier/KNN (k-nearest neighbors algorithm)

#### 2.1.6.1 Logistic Regression

Logistic regression is a widely used algorithm because it is a widely used algorithm and the logic is simple. Logistic regression can solve most dichotomous problems, such as illness, spam, etc. Although it is not the best algorithmic model, the accuracy calculated by this basic model can be used as a reference for more difficult algorithms to verify that it fits better with the analysis of the data se(Pant, A. (2019))t.

To explore whether scaling and data balancing using SMOTE had an effect on the predictions of the model, I ran and evaluated both algorithms before and after scaling and SMOTE for comparison.

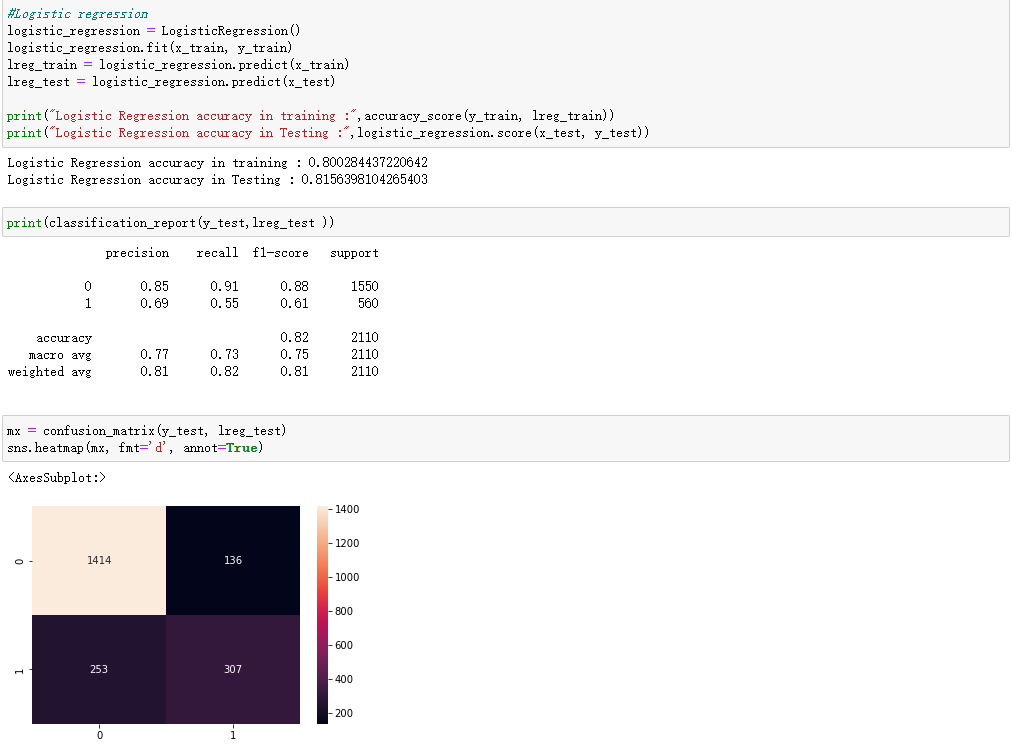


Figure 23 -- before balance/scaling dataset

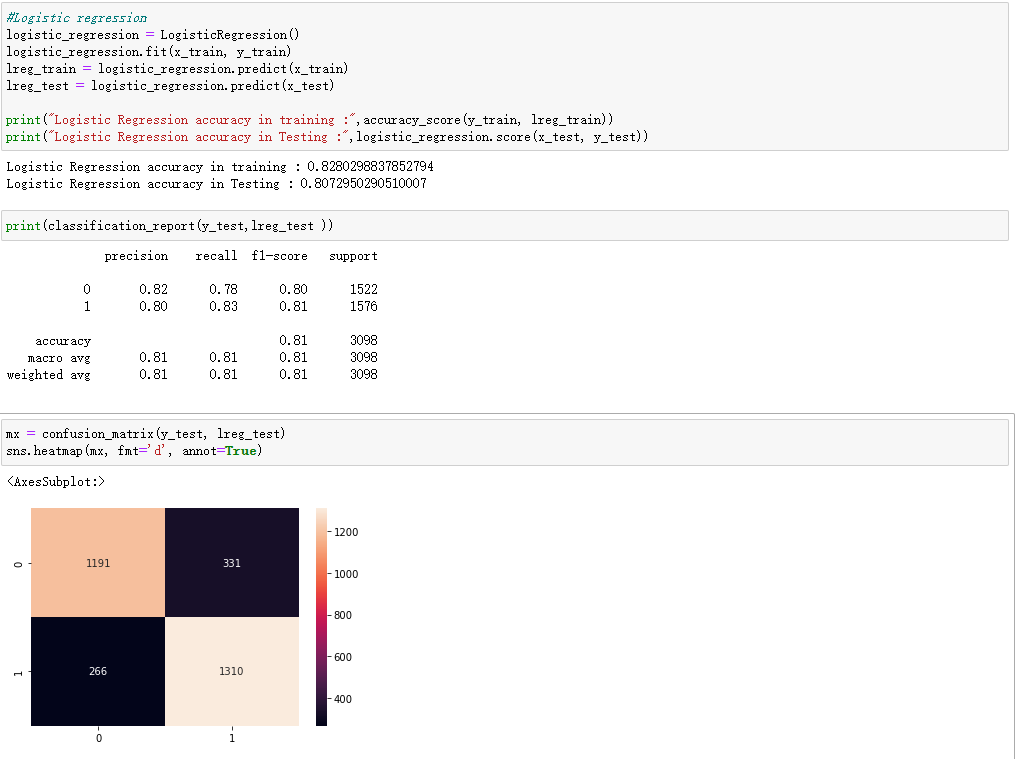


Figure 24 -- after balance/scaling dataset

Based on the comparison between Figure 24 and Figure 25, the overall TRAINING scores have gone up a little. It is worth noting that in figure '1': churn-Yes, because of the Smote manual averaging of the data, the scores for precision and recall and F1 have increased significantly. However, there is a drop in procision from the matrices plot in test and true determination '0' : Churn- No. This also very clearly highlights the 'overfitting' that can result from using logistic regression algorithms - caused by too little training data or data that is too complex for the algorithm to be too simple.

#### 2.1.6.2 Gradient Boosting Classifier

Gradient Boosting is a model that works very well for classification problem datasets - based on decision trees. It has the flexibility to handle all types of data, including discrete and continuous values. Not only that, it has high accuracy readings for its measurements. However, the disadvantage is that it is computationally complex and time consuming(Nelson, D. (2022).

As with the Logistic Regression, we compare the model before and after scaling and SMOTE.

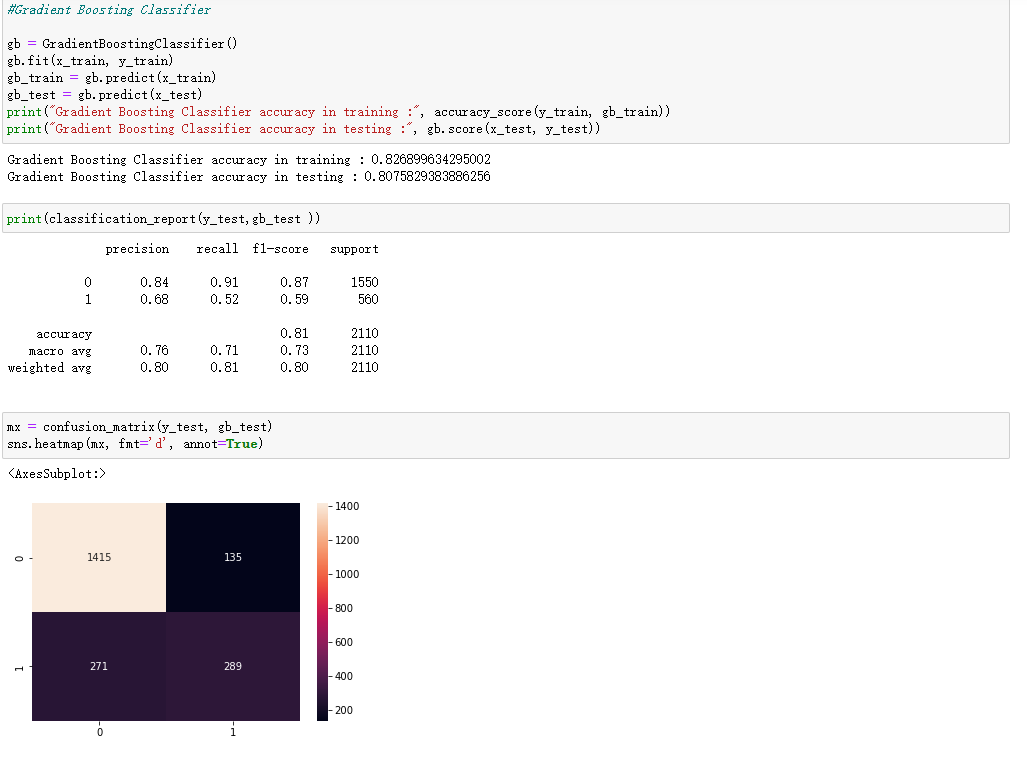


Figure 25 -- before scaling/SMOTE dataset

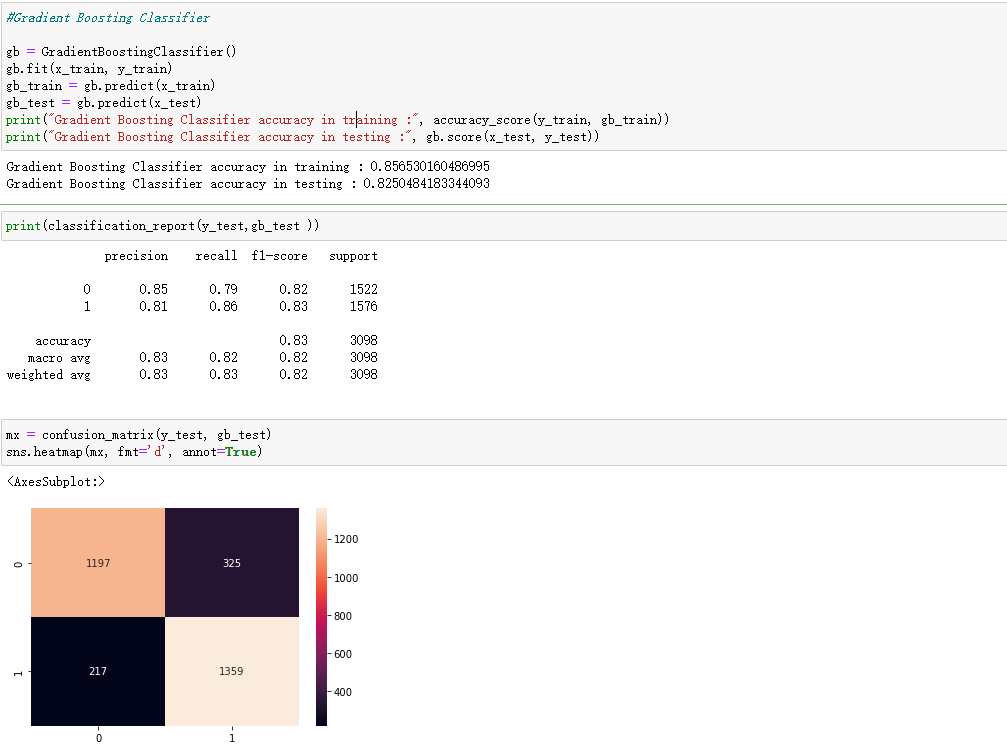


Figure 26 -- after SMOTE/ Scaling dataset

Comparing this in Figure 25/26, we can see that there is a significant drop in the recall value of '0' after scaling and smote. This is because the gradient boosting algorithm focuses on the majority of people rather than the minority. This also causes a drop in the F1 value for '0'. It has to be said, however, that the F1 values from the Marco average (which focuses on the unbalanced dataset prediction scores) show that the unbalanced dataset scores are higher after SMOTE and scaling - this is due to the more balanced data after Smote and scaling.

(https://stats.stackexchange.com/questions/342741/why-am-i-getting-lower-recall-with-boosted-tree-than-decision-tree-on-unbalanced )

#### 2.1.6.3 K-nearest neighbors algorithm (KNN)

KNN is a simple - machine learning algorithm for classification problems that can be used for non-linear classification problems. The principle is to find the smallest k points in a data set based on the distance between points, sorted in increasing order of distance, and to detect the probability of the k points occurring in a category(harrison, onel. (2018).

As with the Logistic Regression, we compare the model before and after scaling and SMOTE.

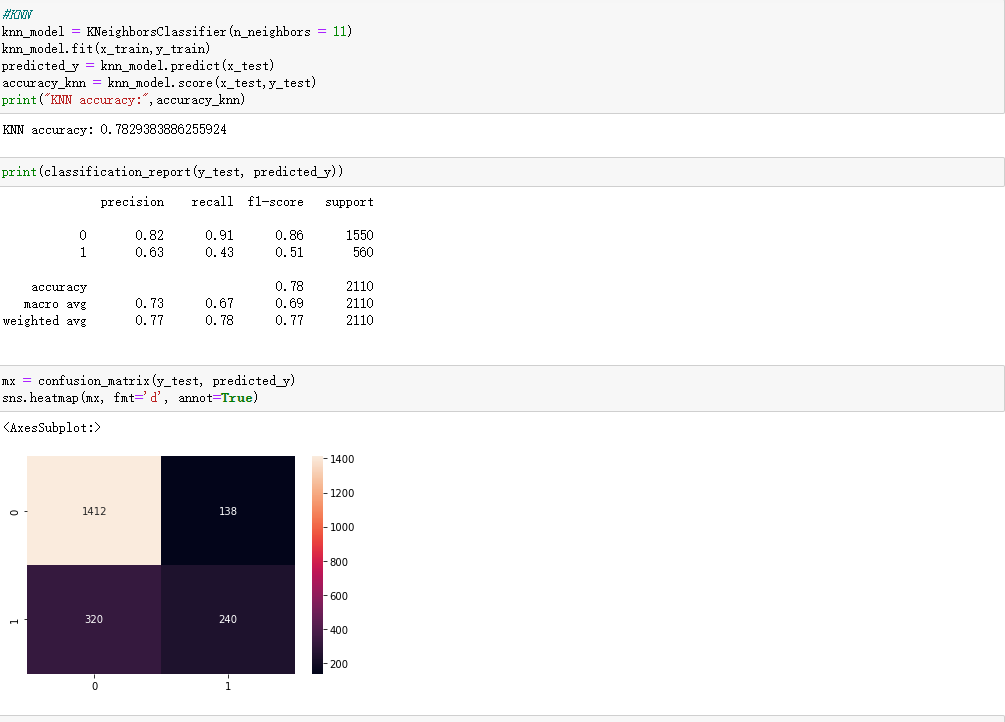


Figure 27 -- before SMOTE/ scaling the dataset

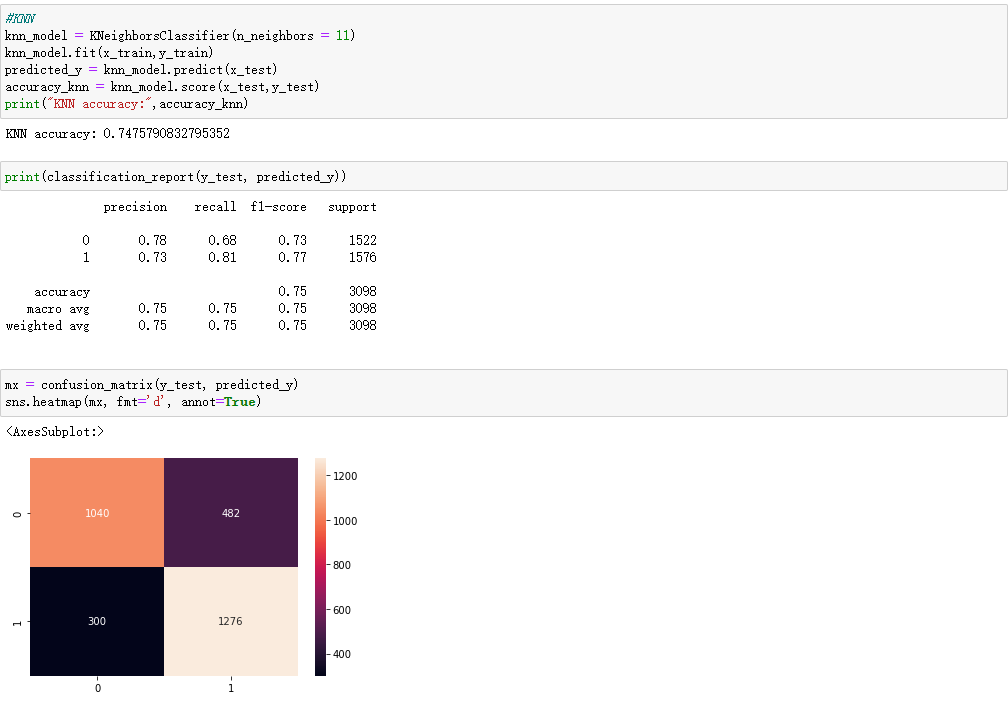


Figure 28 -- after SMOTE/ scaling the dataset

By comparing figures 27 and 28, you can see that smote and Scaling are very effective. It is found from '0 'that the scores of F1 and recall have been greatly improved. But, for the accuracy score have a little bit decrease.

### 2.1.7 Model Turning

Next, we need to optimize these three models.

When tuning, I use GridSearchCV for tuning. It is divided into two parts - grid search and cross validation. Grid search is to find each possibility within the range of parameters, and the parameter with the best performance is the final result. Cross-validation is to evaluate the data set for several times to evaluate the average value, so as to eliminate the adverse effects caused by the imbalance of data division during the single division.

First we turn the logistic regression algorithm model.

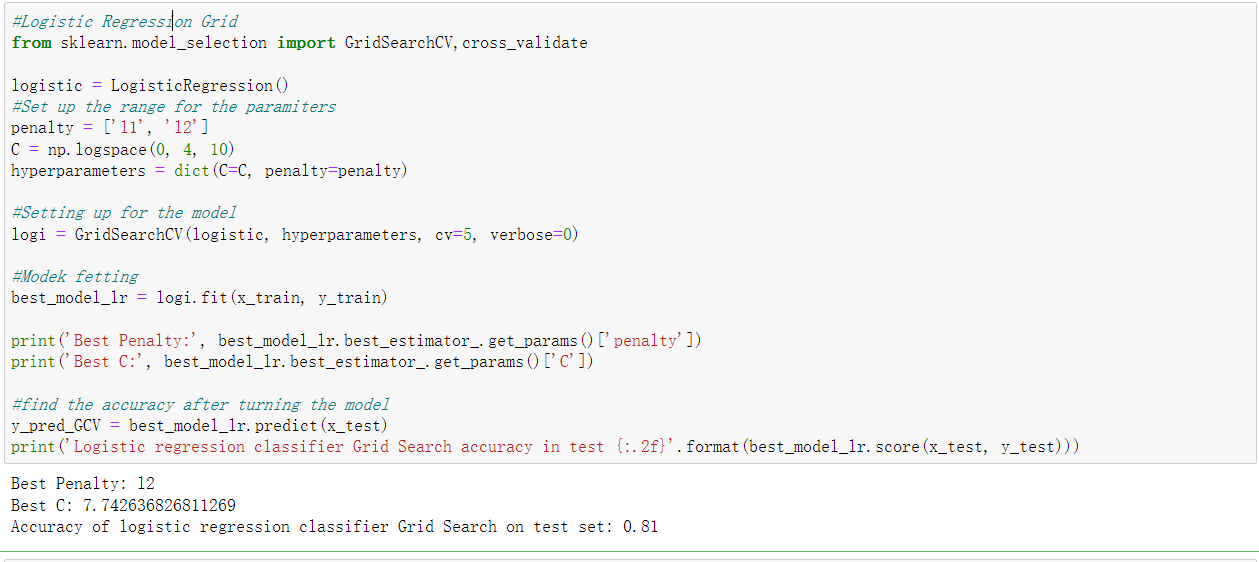


Figure 29 -- turning the logistic regression

I set penlty to 11,12 to find. For parameter C, I set the start point to 0 and the end point to 4. There are 10 elements in total. And then trying to setting up the model,and we will set the split number of cross validation to 5 , then do the fetting.

Finally, we found the best penalty was 12, and the accuracy was 0.81. By comparison with before turning the model, the accuracy number is increase about 0.02.

Secondly, we do the same thing in the Gradient Boosting Classifier

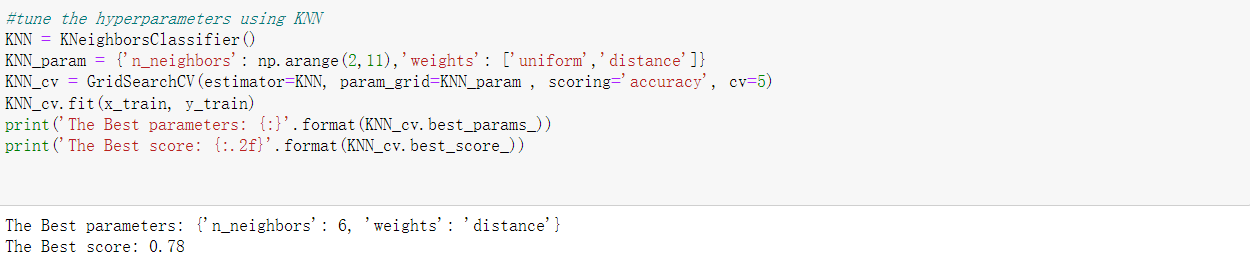


Figure30 -- turning the Gradient Boosting Classifier

First, we need to set the parameters in the bosssting framework, then trying to setting up the model,and we will set the split number of cross validation to 5 , then do the fetting.

In the end, we got an accuracy score of 0.833. By comparison, it was 0.01 percent higher than before turning.

Finally, we turning the KNN model



Fiture 31 -- turning the KNN

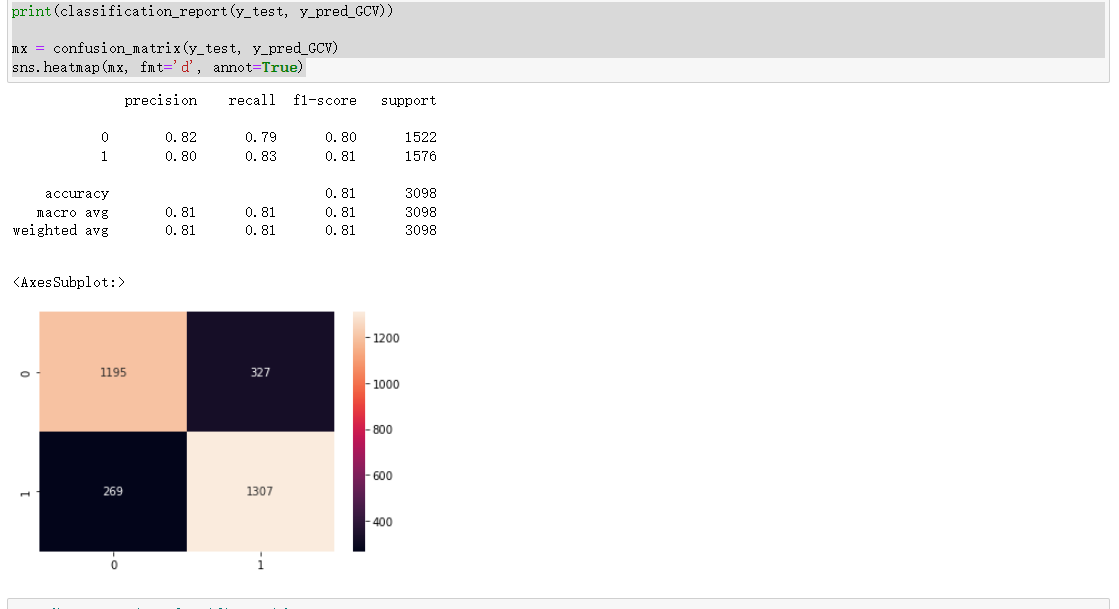
First of all, we set a cycle to find the number of neighbors from 2 to 11, and the weights of weights are set in uniform (uniform weight), where all points in each adjacent region have the same weight, and distance (reciprocal of the weight of power and distance).

Finally, we find that the best parameters is neighbors is 6, and weight is ‘distance’. and finally get the best score is 0.78. compare with the previous one, and find that it almost at the same score.

### 2.1.8 model evaluation

In the final step, we evaluate the data against the optimised model.

First, talk about Logistics regression.



Fiture 32 -- evaluation for logistic regression

As can be seen from the graph, the values for accuracy and recall are 0.79 to 0.83 respectively. the F1 values for 0 and 1 are relatively equal, both being around 0.8 - F1 gives an overall assessment of the model. The F1 values for accuracy, macro mean and weighted mean are all equal - all at 8.1. so the model performs quite well overall(Leung, K. (2022)).

Furthermore, as can be seen from the heat map, the difference between the proportion of his predictions that are correct and the proportion that are actually correct is small - a little more churn\_Yes predictions and a small difference between the proportion of predictions that are correct and the proportion that are actually wrong - a little higher proportion of churn-yes predictions and actually NO at 327.

Next, going to the Gradient Boosting Classifier.

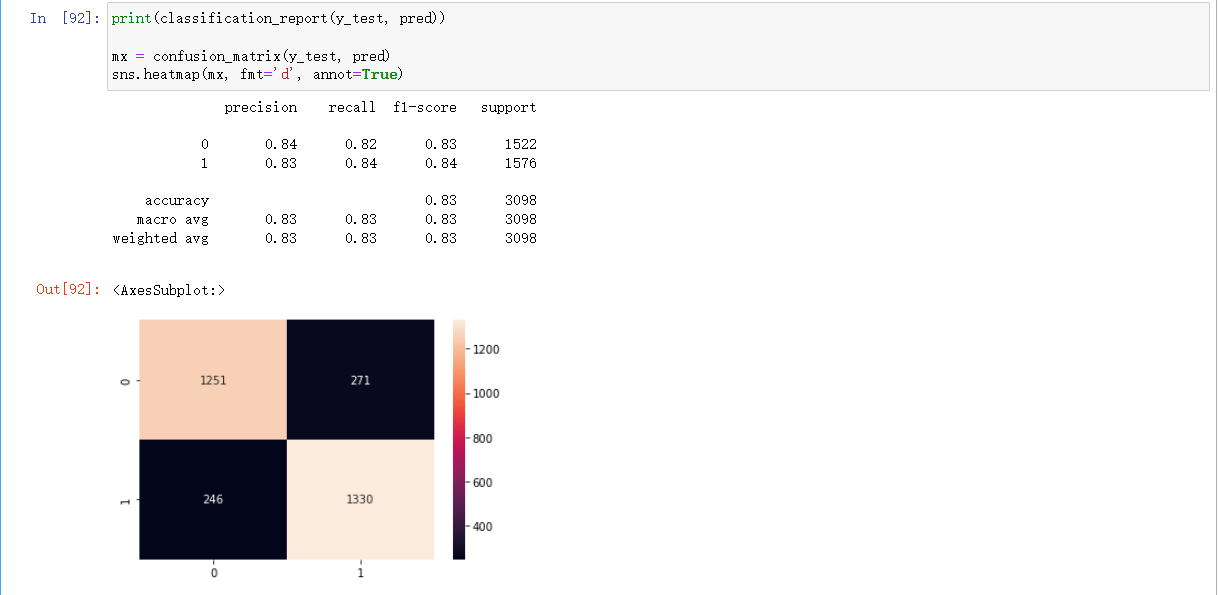


Figure32 -- evaluation for Gradient Boosting Classifier

As we can see from the graphs, the accuracy and recall for both 0 and 1 are between 0.82 and 0.84, and the F1 scores are between 0.83 and 0.83. Not only that, but the F1 scores for accuracy and macro/weighted average are all very average, staying within 08.3 on average, which proves that the overall performance of the model is good and does not deviate too much.

Also, based on the values in the heat map, predicting churn\_yes would be more accurate with a value of 1300, while predicting churn\_yes while actually being wrong would be a little confusing with a value of 270.

Finally, we are going to the KNN model

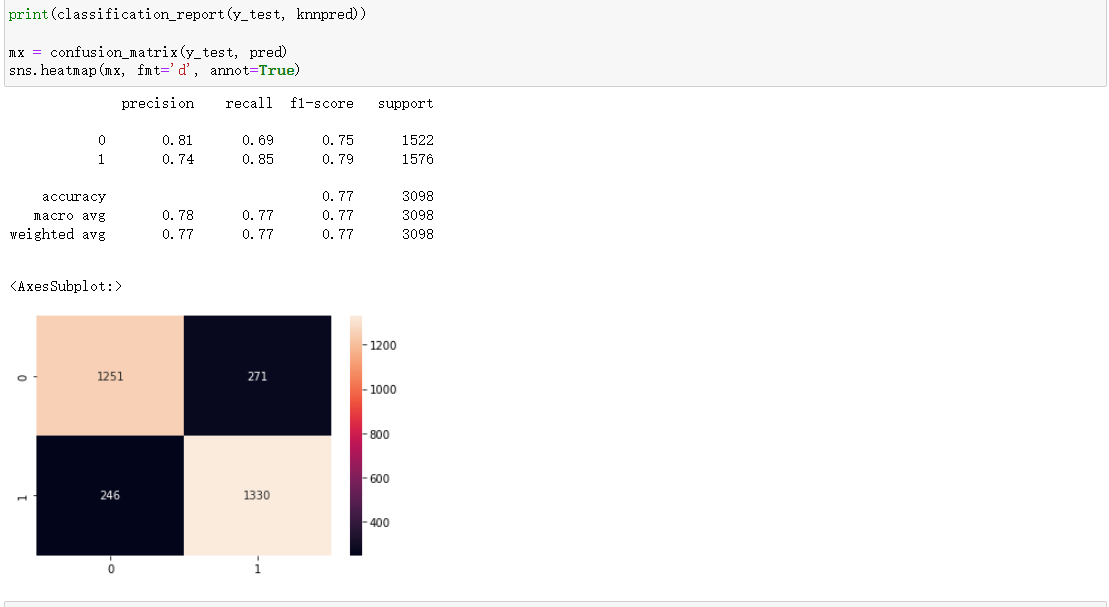


Figure 33 -- evaluation for KNN

As can be seen from the graph, the values for accuracy and recall are 0.69 to 0.85 respectively. the F1 for 0 and 1 are 0.75 and 0.79 respectively. the F1 values for accuracy, macro mean and weighted mean are all equal - all are 0.77. looking at the model as a whole, the model collates performance bias and fluctuates relatively widely.

In addition, the heat map shows that there is little difference between the proportion of his predictions that are correct and the proportion that are actually correct - a little more churn\_Yes predictions and a little less difference between the proportion of predictions that are correct and the proportion that are actually wrong - a little more churn-yes predictions and a little more NO predictions at 271.

So, in general, i think Gradient Boosting Classifier will be more fitful for this dataset.

## Conclusion

In summary, we have successfully used machine learning to successfully process an imbalanced dataset of telecoms customer churn. We used data visualisation to see the relationship between some of the features and Churn and to sort out the possible relationships between the different features. Three different algorithmic models were tested and optimised, and the Gradient Boosting Classifier was chosen as the algorithmic model for this project, and was able to achieve an accuracy of 0.83.

[**word count**:3426 ]

|  |
| --- |
| **Bibliography** |

1. Deshpande, T. (2022, September 27). Telco Churn: EDA|CV Score (85%+)| F1 Score (80%+). Kaggle. [https://www.kaggle.com/code/tanmay111999/telco-churn-eda-cv-score-85-f1-score-80#Algorithm-Results-Table](https://www.kaggle.com/code/tanmay111999/telco-churn-eda-cv-score-85-f1-score-80" \l "Algorithm-Results-Table)
2. Outlier. (n.d.). Wikipedia. <https://en.wikipedia.org/wiki/Outlier>
3. Oracle® Crystal Ball Reference and Examples Guide. (n.d.). Orcale.

https://docs.oracle.com/cd/E57185\_01/CBREG/ch03s02s03s01. html

1. Venmani, a d. (2020, April 16). Python Boxplot – How to Create and Interpret Boxplots (Also Find Outliers and Summarize Distributions). Machine Learning+. <https://www.machinelearningplus.com/plots/python-boxplot/>
2. mcneese, B. (2008, August). Are the Skewness and Kurtosis Useful Statistics? Spcforexcel. <https://www.spcforexcel.com/knowledge/basic-statistics/are-skewness-and-kurtosis-useful-statistics>

6.Seaborn.Countplot. (n.d.). Seaborn. https://seaborn.pydata.org/generated/seaborn.countplot.html

1. Brownlee , J. (2020, June 10). How to Use StandardScaler and MinMaxScaler Transforms in Python. Machine Learning Mastery. <https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/>
2. Brownlee , J. (2021, March 17). SMOTE for Imbalanced Classification with Python. Machine Learning Mastery. <https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/>
3. Pant, A. (2019, January 22). Introduction to Logistic Regression. Towards Datas Cience. <https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148>
4. Nelson, D. (2022, July 21). Gradient Boosting Classifiers in Python with Scikit-Learn. Stack Abuse. <https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148>
5. harrison, onel. (2018, September 10). Machine Learning Basics with the K-Nearest Neighbors Algorithm. Towards Data Science. <https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148>

11.Leung, K. (2022, January 4). Micro, Macro & Weighted Averages of F1 Score, Clearly Explained. Towards Data Science. https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f

|  |
| --- |
| **Appendix A** |

< A suggested checklist for you, for full details please refer to the coursework brief >

1. The following naming convention is used for the Coventry GitHub Repository and Coventry OneDrive

StudentID-Initials-s1

For example, a student Liz Truss whose student ID is 12345678 would name their repository or shared folder as **12345678-LT-s1**

Failing to follow the naming convention may delay the release of marks and feedback for your coursework.

1. **Coventry** GitHub Repository URL **or** **Coventry** OneDrive URL: added to the top of this report
   1. Coventry GitHub Repository includes:

* URL of the selected dataset(s) included in README
* The original selected dataset(s)
* Source-code (.ipynb)
* Demonstration video (.mp4)
  1. Coventry OneDrive folder includes:
* URL of the selected dataset(s) included in a separated text file
* The original selected dataset(s)
* Source-code (.ipynb)
* Demonstration video (.mp4)

1. Source-code added **as text** in Appendix B (below)
2. Submission in the form of a **Word** document. *\*\*Other format is not accepted.*

|  |
| --- |
| **Appendix B** |

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

**import** **plotly.express** **as** **px**

**import** **seaborn** **as** **sns**

**from** **plotly.subplots** **import** make\_subplots

**import** **plotly.graph\_objects** **as** **go**

**import** **warnings**

warnings.filterwarnings('ignore')

**from** **sklearn.preprocessing** **import** LabelEncoder

**from** **sklearn.neighbors** **import** KNeighborsClassifier

**from** **sklearn.ensemble** **import** GradientBoostingClassifier

**from** **sklearn.linear\_model** **import** LogisticRegression

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

**from** **sklearn.metrics** **import** accuracy\_score

**from** **sklearn** **import** metrics

**from** **sklearn.metrics** **import** roc\_curve

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

churn\_dt = pd.read\_csv('WA\_Fn-UseC\_-Telco-Customer-Churn.csv')

churn\_dt.head() #Check the elements present in the dataset

churn\_dt.shape #Check how much data exists in the dataset

churn\_dt.info() #Learn about the elements in each column, including the element name and datatype.

churn\_dt.columns.values #Display the data names that exist in the columns

churn\_dt.notnull() #Check if num exists in each data

churn\_dt.isnull().any() #Check whether there is a null function from the whole

churn\_dt = churn\_dt.drop(['customerID'],axis = **1** )#Remove the 'customerID' in the column, because the customerID has nothing to do with the analysis

churn\_dt.head()#Check if customerID is removed

churn\_dt['TotalCharges'] = pd.to\_numeric(churn\_dt.TotalCharges, errors = 'coerce') #Because totalcharges is not float, change the data type of totalcharges from object to float.

churn\_dt.dtypes

churn\_dt.isnull().sum()#After transformation, the data is checked and found for missing values

churn\_dt[np.isnan(churn\_dt['TotalCharges'])] #Find if there are nan in totalcharges

churn\_dt[churn\_dt['tenure'] == **0**].index #Check if there are too many '0' values in the trnure column, the output shows that we do not have too many missing values

churn\_dt.drop(labels=churn\_dt[churn\_dt['tenure'] == **0**].index, axis=**0**, inplace=True) #Remove the line with the addition value that has no effect on the data

churn\_dt[churn\_dt['tenure'] == **0**].index #Query whether the tenure column still exists, obviously in the output we find that all 0 values have been deleted

churn\_dt.fillna(churn\_dt['TotalCharges'].mean()) #In order to ensure normal analysis, fill the missing value in 'TotelCharges' with the average number

churn\_dt.isnull().sum() #Perform a missing value query for each item again, and obviously find that there are no missing values.

churn\_dt.skew()

# learn from here -- https://www.machinelearningplus.com/plots/python-boxplot/

**def** **boxPlot**(x):

**for** j **in** x:

fig, ax = plt.subplots(**1**, **1**, figsize=(**20**, **5**))

sns.boxplot(data=churn\_dt, x=j, y='Churn')

boxPlot(['MonthlyCharges', 'tenure', 'TotalCharges'])

# box plot for monthlycharges/tenure/totalcharges

#The purpose is to find exceptions number.

#And check whether the data is symmetric, how tightly the data is grouped, and whether the data is skewed, and check the skewness

#totalChargesOutliers = np.where(churn\_dt['TotalCharges'] > 7800)

#churn\_dt.drop(totalChargesOutliers[0], inplace = True)

#try to clear outliers

churn\_dt.skew()

#But it was found to have no effect, so the deletion of outlies will no longer be performed.

# learn from here -- https://seaborn.pydata.org/generated/seaborn.distplot.html

sns.histplot(data=churn\_dt, x="Churn") #Use a histogram to show the proportion of churn

#learn from here -- https://seaborn.pydata.org/generated/seaborn.distplot.html

sns.histplot(data=churn\_dt, x="gender") #Use a bar chart to show the ratio of gender

#learn form here -- https://medium.com/@msjahid/basic-pie-chart-with-python-for-data-visualization-efd7583aa04e

fig, aexs = plt.subplots(**2**,**3**, figsize=(**10**,**5**))

label = ['Male','Female']

gender = churn\_dt["gender"].value\_counts()

aexs[**0**, **0**].pie(gender, autopct='%1.1f%%',labels=label, shadow=True, explode=(**0.05**,**0.05**))

aexs[**0**, **0**].set\_title('Percentage of gender')

label = ['No','Yes']

Churn = churn\_dt["Churn"].value\_counts()

aexs[**0**, **1**].pie(Churn, autopct='%1.1f%%',labels=label, shadow=True, explode=(**0.05**,**0.05**))

aexs[**0**, **1**].set\_title('Percentage of Churn')

label = ['No','Yes']

SeniorCitizen = churn\_dt["SeniorCitizen"].value\_counts()

aexs[**0**, **2**].pie(SeniorCitizen, autopct='%1.1f%%',labels=label, shadow=True, explode=(**0.05**,**0.05**))

aexs[**0**, **2**].set\_title('Percentage of Senior Citizens')

label = ['No','Yes']

Partner = churn\_dt["Partner"].value\_counts()

aexs[**1**, **0**].pie(Partner, autopct='%1.1f%%',labels=label, shadow=True, explode=(**0.05**,**0.05**))

aexs[**1**, **0**].set\_title('Percentage of Partner')

label = ['No','Yes']

Dependents = churn\_dt["Dependents"].value\_counts()

aexs[**1**, **1**].pie(Dependents, autopct='%1.1f%%',labels=label, shadow=True, explode=(**0.05**,**0.05**))

aexs[**1**, **1**].set\_title('Percentage of Dependents')

label = ['No','Yes']

PhoneService = churn\_dt["PhoneService"].value\_counts()

aexs[**1**, **2**].pie(PhoneService, autopct='%1.1f%%',labels=label, shadow=True, explode=(**0.05**,**0.05**))

aexs[**1**, **2**].set\_title('Percentage of PhoneService')

#learn from here --- https://medium.com/@msjahid/basic-pie-chart-with-python-for-data-visualization-efd7583aa04e

fig, aexs = plt.subplots(**2**,**2**, figsize=(**10**,**5**))

label = ['DSL','Fiber optic','No']

InternetService = churn\_dt["InternetService"].value\_counts()

aexs[**0**, **0**].pie(InternetService, autopct='%1.1f%%',labels=label, shadow=True, explode=(**0.05**,**0.05**,**0.05**))

aexs[**0**, **0**].set\_title('Percentage of InternetService')

label = ['Month-to-month','One year','Two year']

Contract = churn\_dt["Contract"].value\_counts()

aexs[**0**, **1**].pie(Contract, autopct='%1.1f%%',labels=label, shadow=True, explode=(**0.05**,**0.05**,**0.05**))

aexs[**0**, **1**].set\_title('Percentage of Contract')

label = ['No','Yes','No internet service']

StreamingTV = churn\_dt["StreamingTV"].value\_counts()

aexs[**1**, **1**].pie(StreamingTV, autopct='%1.1f%%',labels=label, shadow=True, explode=(**0.05**,**0.05**,**0.05**))

aexs[**1**, **1**].set\_title('Percentage of StreamingTV')

label = ['No','Yes','No internet service']

StreamingMovies = churn\_dt["StreamingMovies"].value\_counts()

aexs[**1**, **0**].pie(StreamingMovies, autopct='%1.1f%%',labels=label, shadow=True, explode=(**0.05**,**0.05**,**0.05**))

aexs[**1**, **0**].set\_title('Percentage of StreamingMovies')

#learn from here -- https://seaborn.pydata.org/generated/seaborn.countplot.html

fig, ax = plt.subplots(figsize=(**10**,**8**)) # use countplot to show the relation between gender and churn.

sns.countplot(x='gender',hue='Churn',data=churn\_dt,ax=ax)

# learn from here -- https://plotly.com/python/histograms/

fig = px.histogram(churn\_dt, x="Churn", color="Contract", barmode="group", title="Customer contract")

fig.update\_layout(width=**650**, height=**400**, bargap=**0.2**)

fig.show()

churn\_dt["InternetService"].unique()

churn\_dt[churn\_dt["gender"]=="Female"][["InternetService", "Churn"]].value\_counts()

churn\_dt[churn\_dt["gender"]=="Male"][["InternetService", "Churn"]].value\_counts()

# learn from here -- https://plotly.com/python/graph-objects/

figer = go.Figure()

figer.add\_trace(go.Bar(

x = [['Churn:YES', 'Churn:YES', 'Churn:NO', 'Churn:NO'],

["Male", "Female", "Male", "Female"]],

y = [**633**, **664**,**910** ,**889** ],

name = 'Fiber optic',

))

figer.add\_trace(go.Bar(

x = [['Churn:YES', 'Churn:YES', 'Churn:NO', 'Churn:NO'],

["Male", "Female", "Male", "Female"]],

y = [ **240**,**219**,**992**,**965**],

name = 'DSL',

))

figer.add\_trace(go.Bar(

x = [['Churn:YES', 'Churn:YES', 'Churn:NO', 'Churn:NO'],

["Male", "Female", "Male", "Female"]],

y = [**57**,**56**, **717**,**690**],

name = 'No Internet',

))

#learn from here -- https://seaborn.pydata.org/generated/seaborn.scatterplot.html

sns.scatterplot(data=churn\_dt, x="MonthlyCharges", y="TotalCharges", hue='Churn')#Use scatterplot to show the relationship between mothlycharges and totalcharges

#learn from here -- https://www.naukri.com/learning/articles/heatmap-in-seaborn/

plt.figure(figsize=(**10**,**10**))

sns.heatmap(churn\_dt.corr(),annot=True)#Use heatmat to show the relationship between elements

#learn from here -- https://seaborn.pydata.org/generated/seaborn.kdeplot.html

ax = sns.kdeplot(churn\_dt.MonthlyCharges[(churn\_dt["Churn"] == 'Yes') ],color="Blue", shade = True)

ax = sns.kdeplot(churn\_dt.MonthlyCharges[(churn\_dt["Churn"] == 'No') ],ax =ax, color="Red", shade= True)

ax.legend(["Churn","Not Churn"],loc='upper right')

ax.set\_ylabel('Density')

ax.set\_xlabel('Monthly Charges')

ax.set\_title('Relation between Monthly Charges and Churn')

# From line38 - 46, which is learning from Aula 6006cem week3 solutions - Exercise3

**from** **sklearn.preprocessing** **import** OneHotEncoder

encoder = OneHotEncoder(drop='first', dtype=np.int)

dfn = churn\_dt[['gender','Partner','Dependents','PhoneService','MultipleLines','InternetService','OnlineSecurity','OnlineBackup','DeviceProtection','TechSupport','StreamingTV','StreamingMovies','Contract','PaperlessBilling','PaymentMethod']]

dfn = encoder.fit\_transform(dfn).toarray()

dfn = pd.DataFrame(dfn)

churn\_dt.reset\_index(drop=True, inplace=True)

dfn.reset\_index(drop=True, inplace=True)

churn\_dt = pd.concat([churn\_dt,dfn],axis=**1**)

churn\_dt.drop(['gender','Partner','Dependents','PhoneService','MultipleLines','InternetService','OnlineSecurity','OnlineBackup','DeviceProtection','TechSupport','StreamingTV','StreamingMovies','Contract','PaperlessBilling','PaymentMethod'],axis=**1**

,inplace=True)

churn\_dt.head()

#which is learn from here: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

encoder = LabelEncoder()

churn\_dt['Churn'] = encoder.fit\_transform(churn\_dt['Churn'])

churn\_dt

x = churn\_dt.drop('Churn', axis = **1**)

y = churn\_dt['Churn']

#Divide the data set into training and testing

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=**0.3**)

#which is learning from here:https://www.w3schools.com/python/python\_ml\_logistic\_regression.asp

#Logistic regression model

logistic\_regression = LogisticRegression()

logistic\_regression.fit(x\_train, y\_train)

lreg\_train = logistic\_regression.predict(x\_train)

lreg\_test = logistic\_regression.predict(x\_test)

**print**("Logistic Regression accuracy in training :",accuracy\_score(y\_train, lreg\_train))

**print**("Logistic Regression accuracy in Testing :",logistic\_regression.score(x\_test, y\_test))

#which is learn from here: https://www.cnblogs.com/178mz/p/8558435.html

**print**(classification\_report(y\_test,lreg\_test ))

# print matrix to see the relationship between predict and real

mx = confusion\_matrix(y\_test, lreg\_test)

sns.heatmap(mx, fmt='d', annot=True)

# Gradient Boosting Classifier model

# which is learn from here https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

gbc = GradientBoostingClassifier()

gbc.fit(x\_train, y\_train)

gbc\_train = gbc.predict(x\_train)

gbc\_test = gbc.predict(x\_test)

**print**("Gradient Boosting Classifier accuracy score in training is:", accuracy\_score(y\_train, gbc\_train))

**print**("Gradient Boosting Classifier accuracy score in testing is:", gbc.score(x\_test, y\_test))

**print**(classification\_report(y\_test,gbc\_test ))

mx = confusion\_matrix(y\_test, gbc\_test)

sns.heatmap(mx, fmt='d', annot=True)

#KNN model

#https://www.geeksforgeeks.org/k-nearest-neighbor-algorithm-in-python/

knn\_MD = KNeighborsClassifier(n\_neighbors = **11**)

knn\_MD.fit(x\_train,y\_train)

predi\_y = knn\_MD.predict(x\_test)

acc\_knn = knn\_MD.score(x\_test,y\_test)

**print**("KNN accuracy score is :",acc\_knn)

**print**(classification\_report(y\_test, predi\_y))

mx = confusion\_matrix(y\_test, predi\_y)

sns.heatmap(mx, fmt='d', annot=True)

#Explore whether the data is balanced

**from** **collections** **import** Counter

z = churn\_dt['Churn'].values

counter = Counter(z)

**print**('now',counter)

#Use SMOTE to balance the data

# learn from here：https://blog.csdn.net/dzysunshine/article/details/89046831

**from** **imblearn.over\_sampling** **import** SMOTE

x1 = churn\_dt.drop('Churn', axis = **1**)

y1= churn\_dt['Churn']

counter = Counter(y)

**print**('Before',counter)

smt=SMOTE()

x\_train\_sm, y\_train\_sm = smt.fit\_resample(x1,y1)

counter=Counter(y\_train\_sm)

**print**('After',counter)

# Normalize the data using the MinMax scaler

# Learn from here

**from** **sklearn.preprocessing** **import** MinMaxScaler

sca = MinMaxScaler()

churn\_ft = ['tenure', 'MonthlyCharges', 'TotalCharges']

churn\_dt[churn\_ft] = sca.fit\_transform(churn\_dt[churn\_ft])

**print**('Data after scaling:')

churn\_dt.describe()

#Again, the data is divided into tests and training

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x\_train\_sm, y\_train\_sm, test\_size=**0.3**)

#which is learning from here:https://www.w3schools.com/python/python\_ml\_logistic\_regression.asp

#Logistic regression model

logistic\_regression = LogisticRegression()

logistic\_regression.fit(x\_train, y\_train)

lreg\_train = logistic\_regression.predict(x\_train)

lreg\_test = logistic\_regression.predict(x\_test)

**print**("Logistic Regression accuracy in training :",accuracy\_score(y\_train, lreg\_train))

**print**("Logistic Regression accuracy in Testing :",logistic\_regression.score(x\_test, y\_test))

#which is learn from here: https://www.cnblogs.com/178mz/p/8558435.html

**print**(classification\_report(y\_test,lreg\_test ))

# print matrix to see the relationship between predict and real

mx = confusion\_matrix(y\_test, lreg\_test)

sns.heatmap(mx, fmt='d', annot=True)

# Gradient Boosting Classifier model

# which is learn from here https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

gbc = GradientBoostingClassifier()

gbc.fit(x\_train, y\_train)

gbc\_train = gbc.predict(x\_train)

gbc\_test = gbc.predict(x\_test)

**print**("Gradient Boosting Classifier accuracy score in training is:", accuracy\_score(y\_train, gbc\_train))

**print**("Gradient Boosting Classifier accuracy score in testing is:", gbc.score(x\_test, y\_test))

**print**(classification\_report(y\_test,gbc\_test ))

mx = confusion\_matrix(y\_test, gbc\_test)

sns.heatmap(mx, fmt='d', annot=True)

#KNN model

#https://www.geeksforgeeks.org/k-nearest-neighbor-algorithm-in-python/

knn\_MD = KNeighborsClassifier(n\_neighbors = **11**)

knn\_MD.fit(x\_train,y\_train)

predi\_y = knn\_MD.predict(x\_test)

acc\_knn = knn\_MD.score(x\_test,y\_test)

**print**("KNN accuracy score is :",acc\_knn)

**print**(classification\_report(y\_test, predi\_y))

mx = confusion\_matrix(y\_test, predi\_y)

sns.heatmap(mx, fmt='d', annot=True)

#Logistic Regression Grid

#learn from here -- https://www.kaggle.com/code/enespolat/grid-search-with-logistic-regression

**from** **sklearn.model\_selection** **import** GridSearchCV,cross\_validate

logistic = LogisticRegression()

#Set up the range for the paramiters

penalty = ['l1', 'l2']

C = np.logspace(**0**, **4**, **10**)

hyperparameters = dict(C=C, penalty=penalty)

#Setting up for the model

logi = GridSearchCV(logistic, hyperparameters, cv=**5**, verbose=**0**)

#Model fetting

best\_model\_lr = logi.fit(x\_train, y\_train)

**print**('Best Penalty:', best\_model\_lr.best\_estimator\_.get\_params()['penalty'])

**print**('Best C:', best\_model\_lr.best\_estimator\_.get\_params()['C'])

#find the accuracy after turning the model

y\_pred\_GCV = best\_model\_lr.predict(x\_test)

**print**('Logistic regression classifier Grid Search accuracy in test {:.2f}'.format(best\_model\_lr.score(x\_test, y\_test)))

**print**(classification\_report(y\_test, y\_pred\_GCV))

mx = confusion\_matrix(y\_test, y\_pred\_GCV)

sns.heatmap(mx, fmt='d', annot=True)

#learn from here -- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

#Gradient Boosting Classifier Grid

# hyperparameters dictionary

#Pass in the parameters.

param\_GB = {

'n\_estimators':range(**20**,**81**,**10**),

'max\_depth':range(**5**,**16**,**2**),

'min\_samples\_split':range(**200**,**1001**,**200**),

'min\_samples\_leaf':range(**30**,**71**,**10**)

}

#Import parameters into GridSearchCV

cv = GridSearchCV(gbc,param\_GB,cv=**5**, n\_jobs= -**1**)

#Model fetting

cv.fit(x\_train, y\_train)

#Get the accuracy score

pred=cv.predict(x\_test)

acc = accuracy\_score(y\_test,pred)

**print**('accuracy\_score',acc)

Acc\_GB = acc

**print**(classification\_report(y\_test, pred))

mx = confusion\_matrix(y\_test, pred)

sns.heatmap(mx, fmt='d', annot=True)

# learn from here -- https://machinelearningknowledge.ai/knn-classifier-in-sklearn-using-gridsearchcv-with-example/

#tune the hyperparameters using KNN

KNN = KNeighborsClassifier()

KNN\_param = {'n\_neighbors': np.arange(**2**,**11**),'weights': ['uniform','distance']}

KNN\_cv = GridSearchCV(estimator=KNN, param\_grid=KNN\_param , scoring='accuracy', cv=**5**)

KNN\_cv.fit(x\_train, y\_train)

**print**('The Best parameters: {:}'.format(KNN\_cv.best\_params\_))

**print**('The Best score: {:.2f}'.format(KNN\_cv.best\_score\_))

knnpred=KNN\_cv.predict(x\_test)

**print**(classification\_report(y\_test, knnpred))

mx = confusion\_matrix(y\_test, pred)

sns.heatmap(mx, fmt='d', annot=True)