# CUSTOMER CHURN PREDICTION USING MACHINE LEARNING

In this blog-post, I will go through the whole process of creating a machine learning model from loading the dataset to saving the best model on the customer churn dataset from IBM Sample Data Sets. It provides information on the churn rate of customers, summarized according to Age, Tenure, Contract, Monthly charges, Total charges and other features.

## **What is Customer Churn?**

Customer churn is when a company’s customers stop doing business with that company. Businesses are very keen on measuring churn because keeping an existing customer is far less expensive than acquiring a new customer. New business involves working leads through a sales funnel, using marketing and sales budgets to gain additional customers. Existing customers will often have a higher volume of service consumption and can generate additional customer referrals.

## **Why is Customer Retention so important?**

There are some excellent reasons to focus on customer retention:

* It’s up to five times cheaper to retain your current customers than it is to acquire new ones.
* The probability of making a sale to an existing customer is 60-70%
* Retaining your current customers increases word-of-mouth recommendations and loyalty

and many more.

Customer retention can suffer when you attract new customers and grow quickly but struggle to implement a strong customer service strategy as a foundation to support that growth.

## **Customer Retention strategies:**

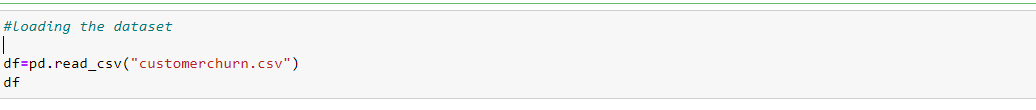
Customer retention can be achieved with good customer service and products. But the most effective way for a company to prevent attrition of customers is to truly know them. The vast volumes of data collected about customers can be used to build churn prediction models. Knowing who is most likely to defect means that a company can prioritise focused marketing efforts on that subset of their customer base.

Preventing customer churn is critically important to the telecommunications sector, as the barriers to entry for switching services are so low.

## **Importing libraries:**

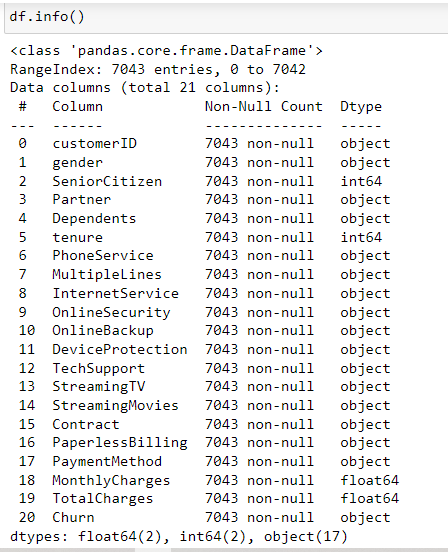


## **Getting the data:**

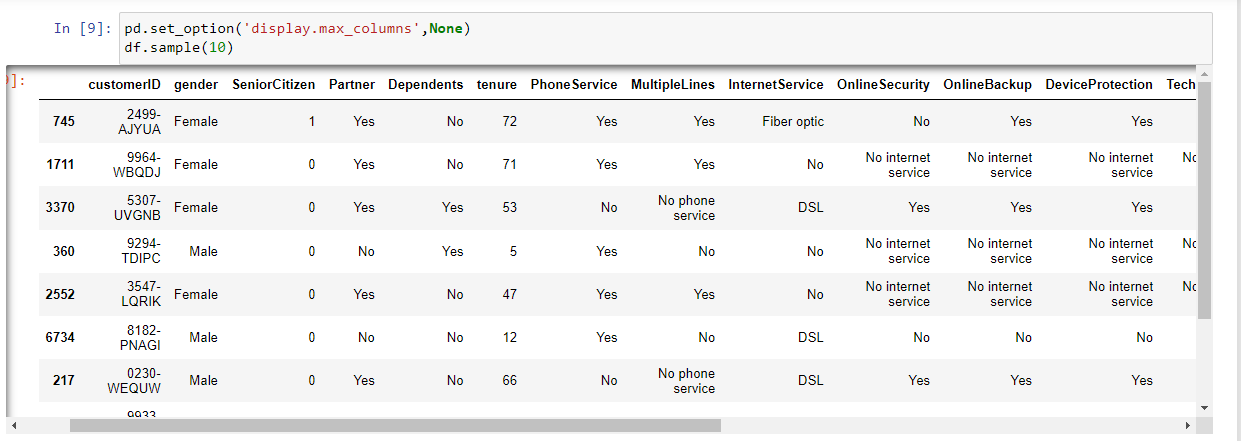


## **Data Analysis and Preprocessing:**

Let’s have a look at the data:



A sample from the dataset:



**The dataset has 7043 examples and 20 features + the target variable (Churn)**. 2 of the features are floats, 2 are integers and 17 are objects. Below I have listed the variables with a short description:

**CustomerID:** A unique ID that identifies each customer.

**Gender:** The customer’s gender: Male, Female

**SeniorCitizen:** Indicates if the customer is 65 or older: Yes, No

**Partner:** Indicates if the customer has a partner: Yes, No

**Dependents:**Indicates if the customer lives with any dependents: Yes, No. Dependents could be children, parents, grandparents, etc.

**tenure:** Indicates the total amount of months that the customer has been with the company.

**PhoneService:** Indicates if the customer subscribes to home phone service with the company: Yes, No

**MultipleLines:** Indicates if the customer subscribes to multiple telephone lines with the company: Yes, No

**InternetService:** Indicates if the customer subscribes to Internet service with the company: No, DSL, Fiber Optic.

**OnlineSecurity:** Indicates if the customer subscribes to an additional online security service provided by the company: Yes, No

**OnlineBackup:** Indicates if the customer subscribes to an additional online backup service provided by the company: Yes, No

**DeviceProtection:** Indicates if the customer subscribes to an additional device protection plan for their Internet equipment provided by the company: Yes, No

**TechSupport:** Indicates if the customer subscribes to an additional technical support plan from the company with reduced wait times: Yes, No

**StreamingTV:** Indicates if the customer uses their Internet service to stream television programing from a third party provider: Yes, No. The company does not charge an additional fee for this service.

**StreamingMovies:** Indicates if the customer uses their Internet service to stream movies from a third party provider: Yes, No. The company does not charge an additional fee for this service.

**Contract:** Indicates the customer’s current contract type: Month-to-Month, One Year, Two Year.

**PaperlessBilling:** Indicates if the customer has chosen paperless billing: Yes, No

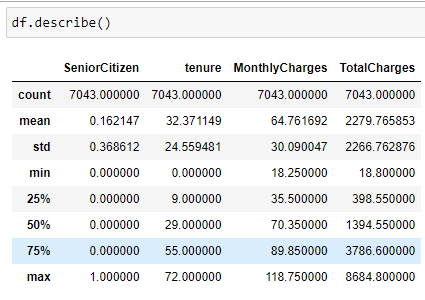
**PaymentMethod:** Indicates how the customer pays their bill: Electronic check, Bank transfer (automatic), Credit Card (automatic), Mailed check

**MonthlyCharge:** Indicates the customer’s current total monthly charge for all their services from the company.

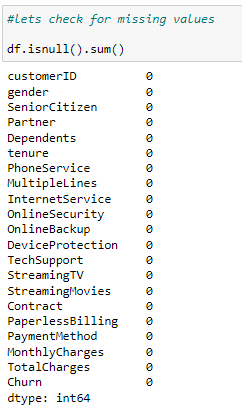
**TotalCharges:** Indicates the customer’s total charges, calculated to the end of the quarter specified above.

**Churn:** Customers who left within the last month: Yes, No

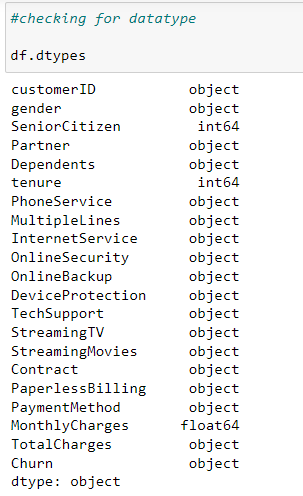
Describing the dataset further:



Now, first thing that I would do is to check for empty or null values, like if they are present in the dataset.

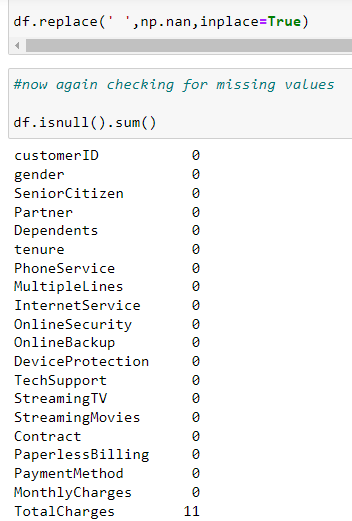


So as we can see that the dataset is showing zero empty values for every variable. Now, let’s look out for data-type of each variable for a better understanding of the dataset:



We can see that variable ‘TotalCharges’ has been assigned ‘object’ datatype instead of ‘float64’ , as it has float values. So, we must convert its datatype from ‘object’ to ‘float64’.

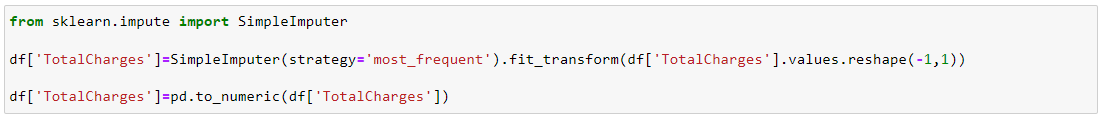
Upon checking into ‘TotalCharges’ before converting its datatype I found that this variables have some empty strings present. In order to convert its datatype first we must replace these empty strings. So, I replaced those empty strings with numpy NaN value and again checked for the count of now null values.



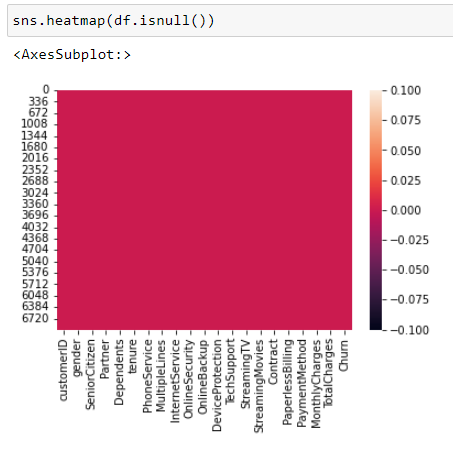
‘TotalCharges’ now has 11 null values. Before converting its datatype, I am going to impute those null values using SimpleImputer from sklearn.

As the variable is of ‘object’ datatype, SimpleImputer with strategy as ‘most\_frequent’ which represents the mode of the data, should be used.

After that I converted the variable ‘TotalCharges’ to ‘float64’ using pandas ‘to\_numeric’ function.



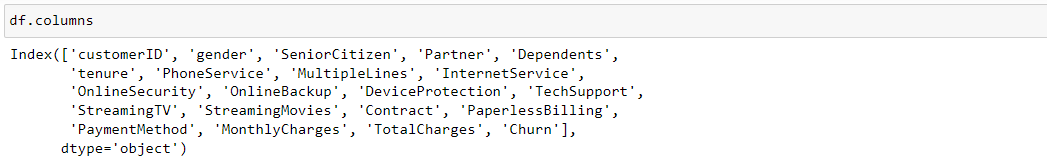
Again checking for null values:



All the null values has been replaced. Now we can move towards exploratory data analysis (EDA).

Since, the preprocessing hasn’t been completed yet, I am going to perform EDA first and just after that I am going to do the left preprocessing i.e. the conversion of ‘object’ datatype variables to integer datatype.

## **Exploratory Data Analysis:**

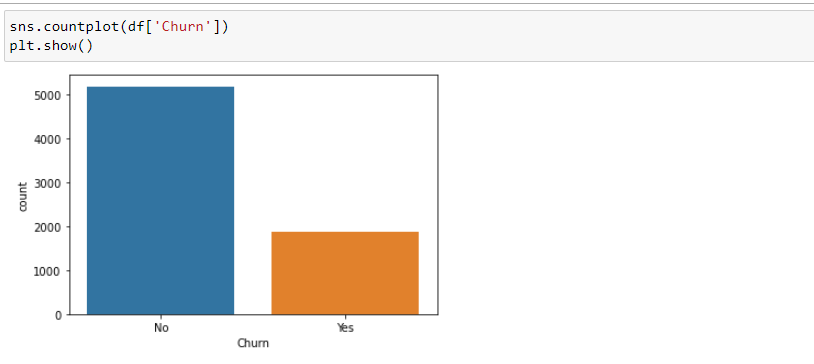


Above you can see the 2 features + the target variable (Churn). What do you think which variables could contribute to the churn rate.

I think to me it would make sense if everything except ‘customerID’, ‘gender’, ‘partner’ and some of the services provided by the providers would be correlated with churn rate.

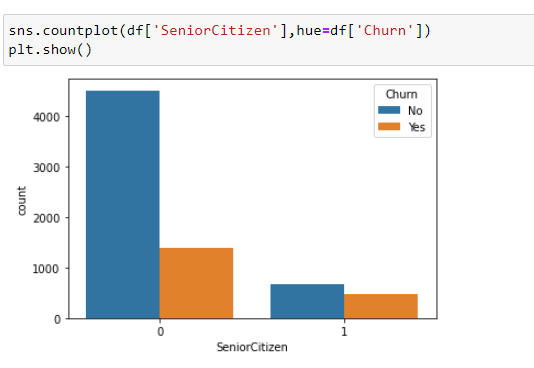
For, the EDA I am going to plot countplot using seaborn library for the ‘object’ datatype variables and for the numerical variables I am going to use catplot with the target(Churn) variable.

1. Churn:



We can see that the labels or classes of the target variable has a imbalanced distribution. Label ‘No’ of Churn has more than twice value count than ‘Yes’ Churn label.

1. SeniorCitizen and Churn:

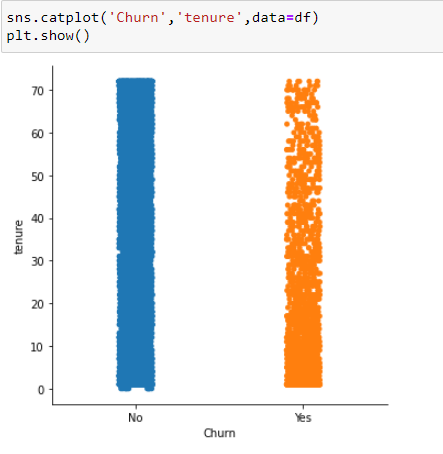


You can see that SeniorCitizen must be correlated with the target variable. Mostly customers are below age of 65.

But the thing is that senior citizens i.e. the customers aged above 65 has a higher rate of churn than the customers below age 65.

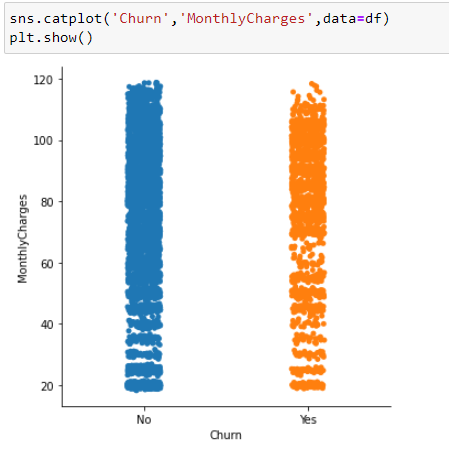
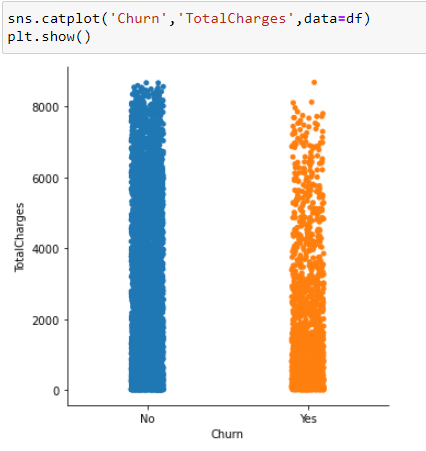
This is why the correlation should be good with the target variable.

1. Tenure and Churn:



From the above catplot it can be concluded that while new customers i.e. of tenure below 20 months have higher rate of churn, old customers i.e. of tenure more than 40 months have reasonably lesser rate of churn.

1. MonthlyCharges , TotalCharges , Churn:

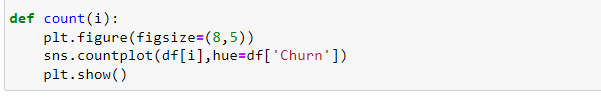
 

While higher monthly charges have higher churn rate, higher total charges have a lower churn rate.

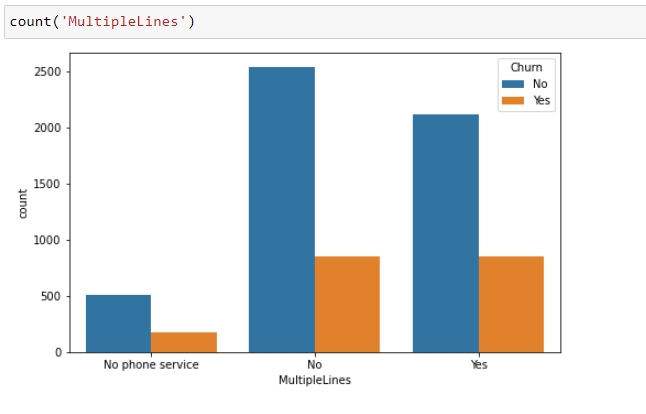
For some lower ranges of monthly charges data for both the labels of target variable is distributed well to some extent.

1. MultipleLines and Churn:

So, as I was going through the EDA I created a user defined function that plots countplot for object datatype variable with Churn as hue.

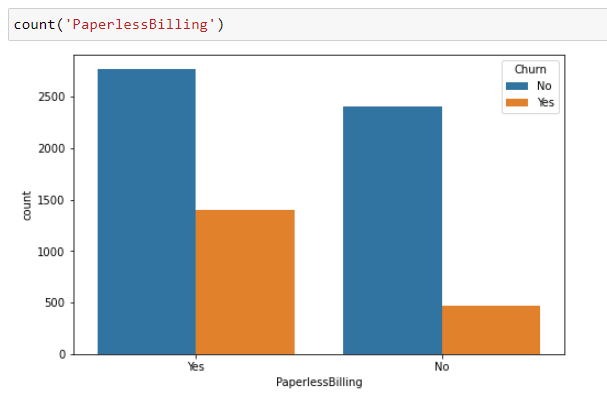


So,



From this you can see that most customers do not have multiple telephone lines. Moreover for customers with multiple telephone lines churn rate is higher than customers with no multiple telephone lines.

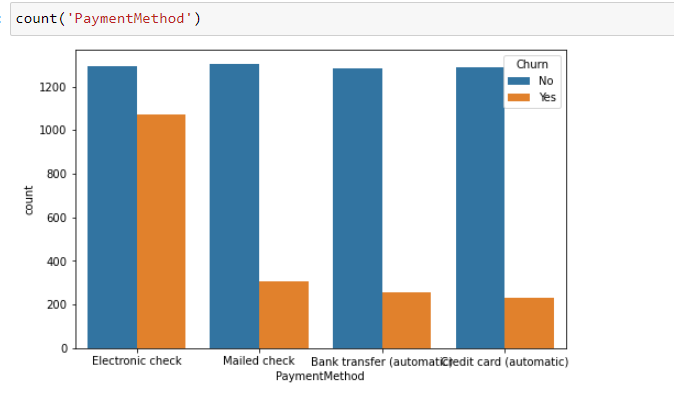
1. PaperlessBilling and Churn:



Both the classes of the above variable have reasonable count. Though more customers prefer paperless billing these customers have found to be with higher churn rate.

Churn rate for no paperless billing customers is exceptionally low.

1. PaymentMethods and Churn:



From here you can see that the data is well distributed for the different classes of above variable.

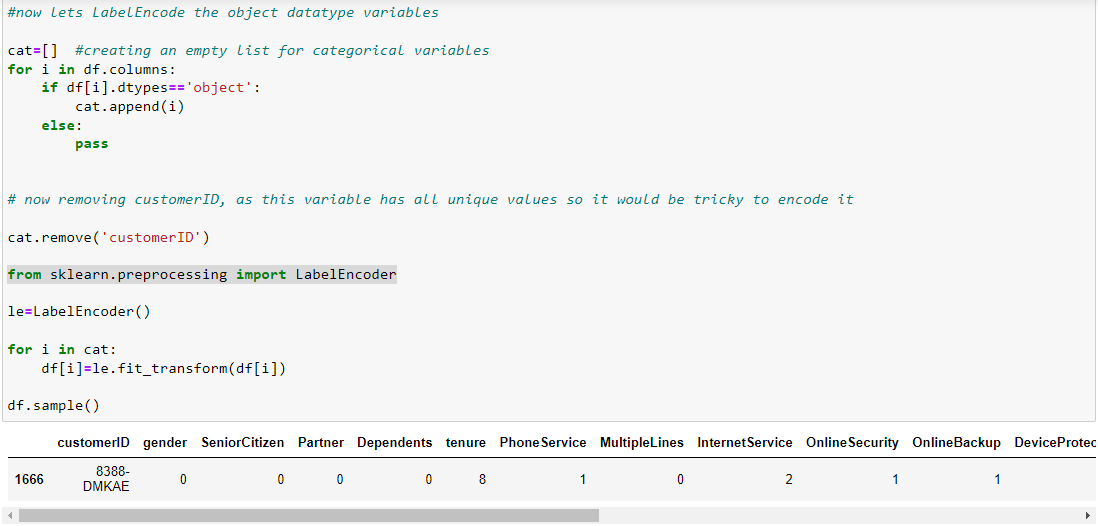
Electronic check is what most customers used based on the dataset but it also has an exceptionally high churn rate. For all the other classes of this variable churn rate is low and almost similar.

This variable seems to be correlated with the target variable.

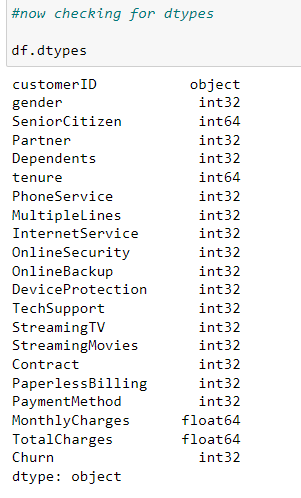
So, now as the EDA has been done before checking for correlation matrix and describing our dataset further conversion of categorical variables should be done.

We have already dealt with datatype conversion of one variable and have handled the missing values too, I am going to create a list of categorical variables present in the dataset and I will drop ‘customerID’ from the list and later from the dataset too, because it shouldn’t contribute to a customer churn probability.

Then I am iterating the list using for loop and passing the categorical variables through LabelEncoder of sklearn,

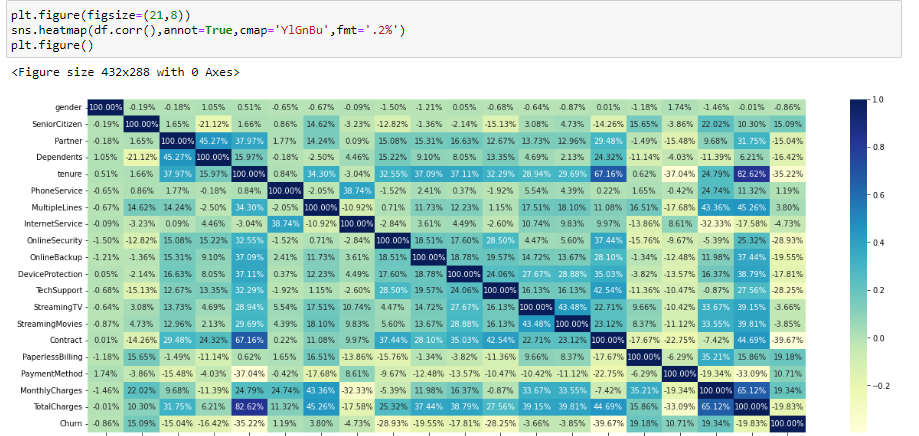


which gives us:



## **Correlation matrix:**

Correlation matrix with respect to the target variable:

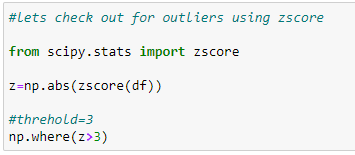


From the above matrix we can conclude that:

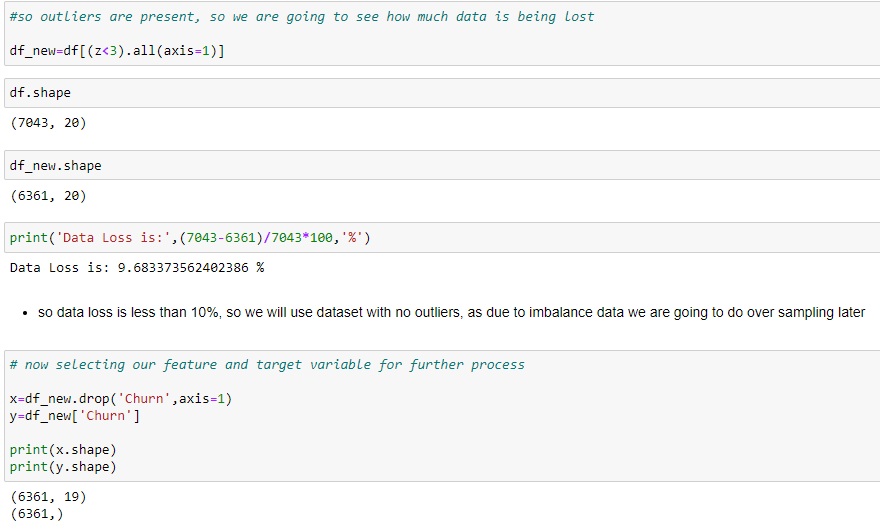
‘MonthlyCharges’, ‘PaperlessBilling’, ‘SeniorCitizen’ have fine correlation bond with the target variable ‘Churn’.

While ‘MultipleLines’ and ‘PhoneServices’ have very weak correlation bond with tha target variable, all the other variables that are left have negative correlation bond with the target variable ‘Churn’.

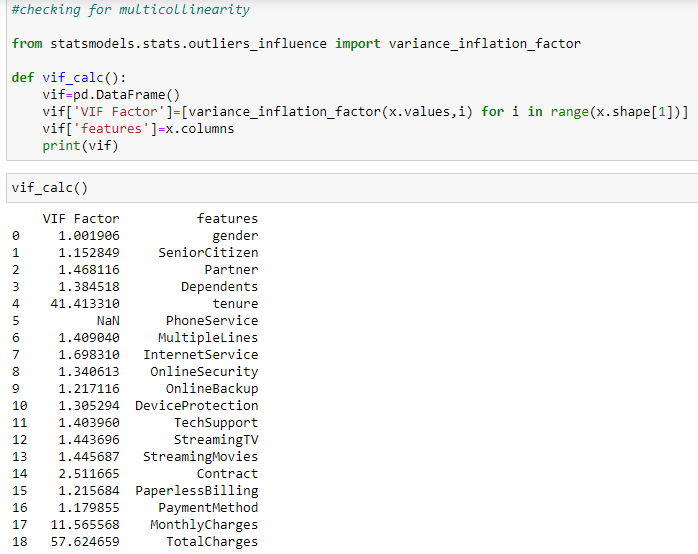
Now before selecting our feature and target variables , I am going to check for outliers using scipy zscore.



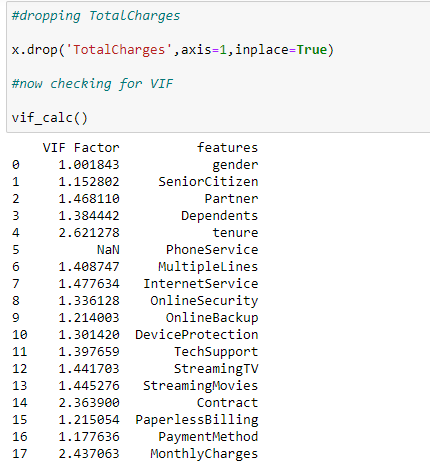
Now, I am going to check the data loss percent and if the data loss is lower than 10% I am going to use the dataset without outliers for further process.



Now as we have selected the variables I am going to check for multicollinearity between the feature variables using variance inflation factor (VIF).

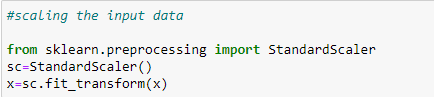


Three variables have VIF greater than 10, which means multicollinearity is present among the feature variables. So I am going to drop the feature variable with high VIF and weak correlation with the target variable and again check for VIF.



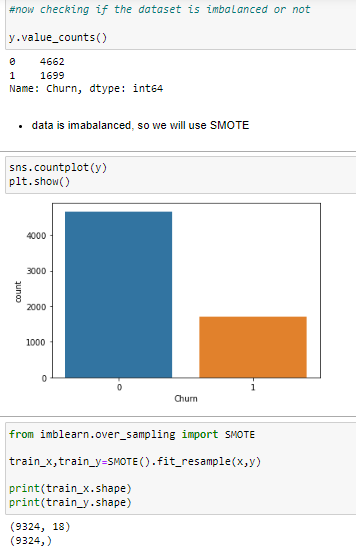
From this you can see that the VIF has been reduced to below five for all feature variables, thus, reduction in multicollinearity.

Now I am going to scale the data using StandardScaler for better results before passing it into the algorithms.



Now, as we have already seen that the target variable has imbalanced class, if we continue with that the dataset might give us poor recall. To handle this I am going to do over sampling using SMOTE. SMOTE would create equal number of counts for both the classes of target variable.

So it is giving us:



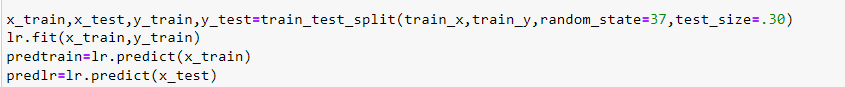
The imbalanced data has been treated now we can process with the algorithms.

## **Building Machine Learning Models:**

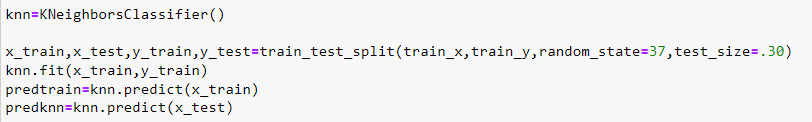
Now we will train several Machine Learning models and compare their results. As the problem has a target variable with two labels or classes this is a binary classification problem.

We are going to compare accuracy score, errors and later cross validation and ROC curve and AUC to find the best model.

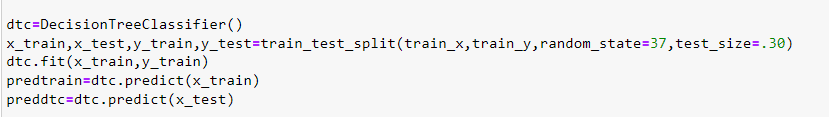
### **Logistic Regression:**



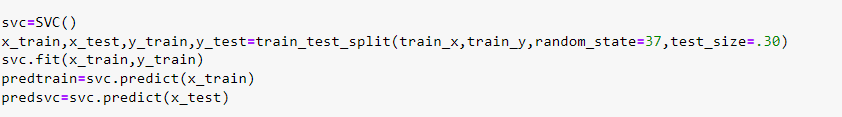
### **K Nearest Neighbor:**



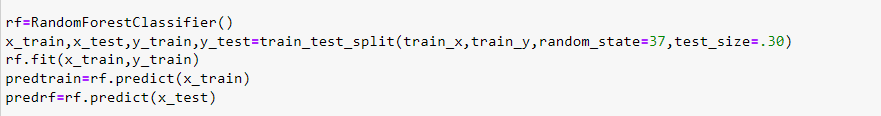
### **Decision Tree:**



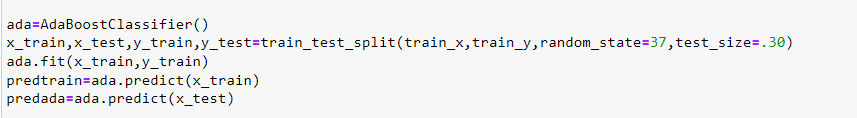
### **Support Vector:**



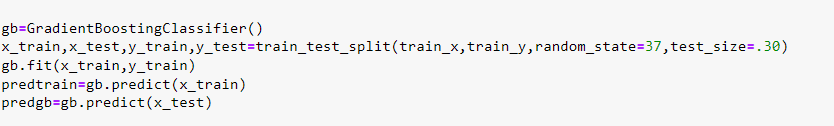
### **Random Forest:**



## **Ada Boost:**



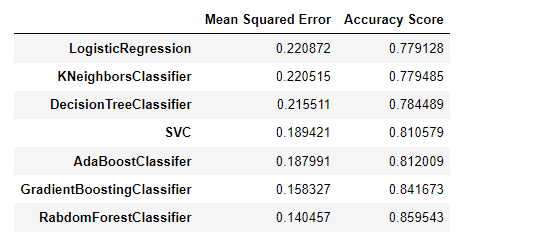
## **Gradient Boosting:**



## **Best Model:**

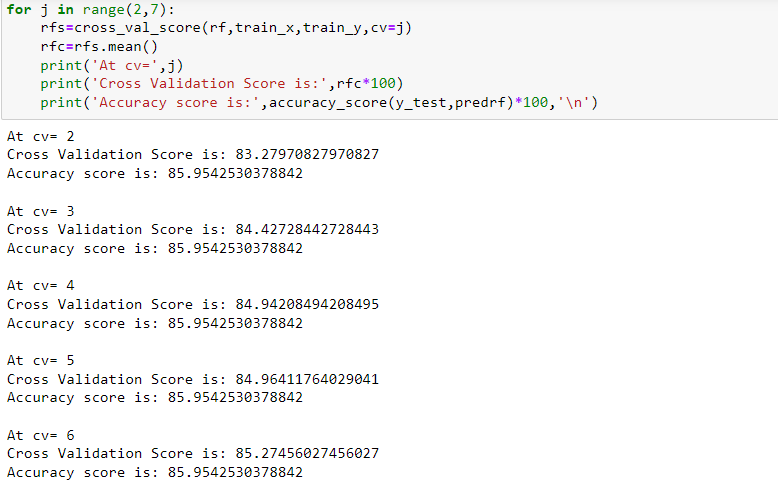


Giving us:



As we can see, the Random Forest Classifier goes on the first place with highest accuracy score and lowest mean squared error among all the other trained models. But first, let us check, how random-forest performs, when we use cross validation.

So I am going to compute the score in a smaller range to see how the model is working.

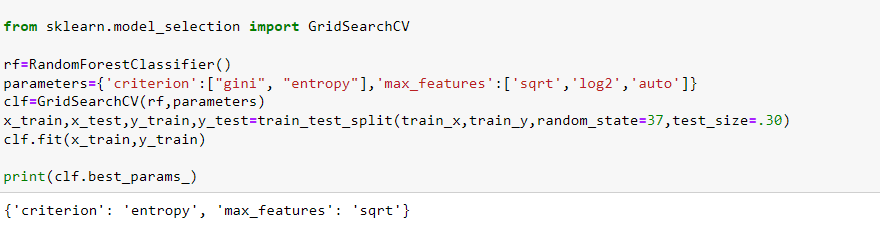


As you can see the accuracy score of the model is coming out to be around the cross validation score with a tiny deviation the model is performing really well.

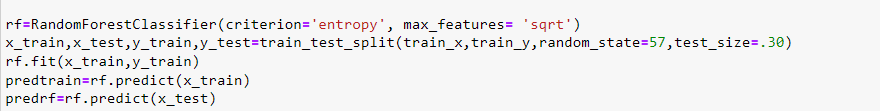
Since Random Forest is an easy to use model, we will try to increase its performance in the following section.

## **Hyperparameter Tuning:**

Below you can see the code of the hyperparamter tuning for the parameters ‘criterion’ and ‘max features’ using Grid Search CV. As the code takes very long time to run, I have used two parameters only.

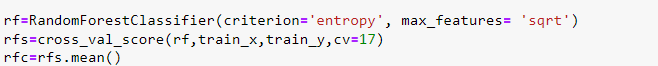


As you can see the best parameter values has been founded. After finding the best suited parameters I put them through the algorithm and found a random state with a good accuracy score.

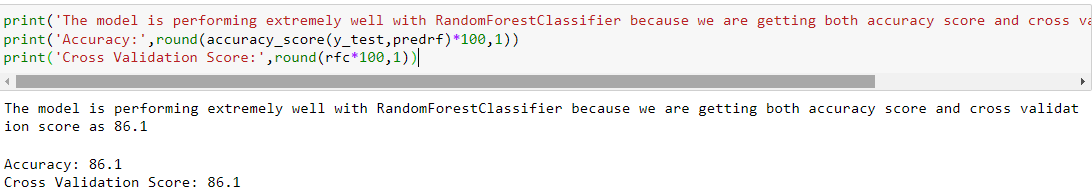


## **Evaluation:**

Now I am going to check for cross validation score with tuned parameters, just to see how the model is working. So, I iterated the CV for a small range, to find the best value of CV that is most close the accuracy score of the model,

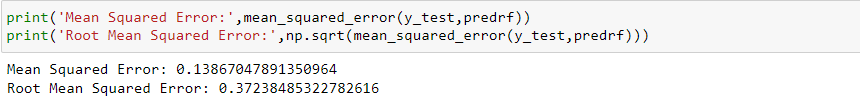


turns out:



We got both the scores similar up to one decimal place.

Now checking for the errors:

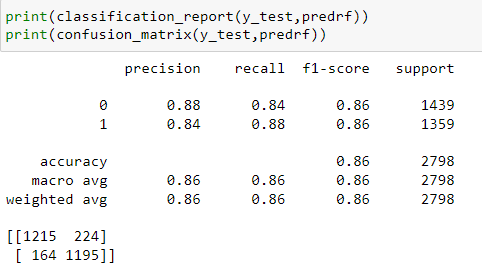


As you can see the errors have been further reduced after hyperparameter tuning. The more the errors are close to zero the better the model is performing.

## **Further Evaluation:**

Now that we have a proper model, we can start evaluating its performance in a more accurate way.

### Confusion Matrix and Classification Report:



**Confusion Matrix:**

The first row is about the not-churn-predictions: 1215 customers were correctly classified as not churn (called true negatives) and 224 were wrongly classified as not churn (false positives).

The second row is about the churn-predictions: 164 customers were wrongly classified as churn (false negatives) and 1195 were correctly classified as survived (true positives).

**Classification Report:**

**Precision and Recall and F-Score:**

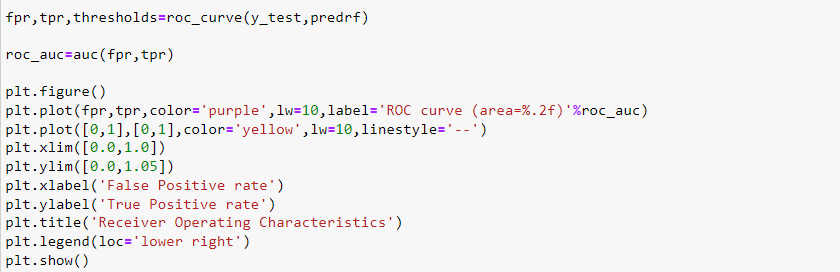
* The model predicts 86% of the time, a customer churns correctly (precision).
* The recall tells us that it predicted the churn of 86 % of the people who actually churn.
* For a binary classification clearly, the higher the f1-score the better, with 0 the worst possible and 1 being the best.

Our model is giving f1-score of 0.86 which is certainly closer to 1.

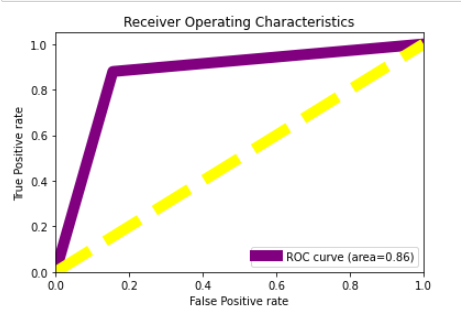
### ROC Curve and AUC SCORE**:**

ROC AUC is used to evaluate and compare your binary classifier in another way and AUC Score is basically the corresponding score to area under curve (AUC).

As the ROC Curve is a plot for true positive rate (recall) against false positive rate I plotted the ROC Curve and placed AUC score in the figure only,



which came up like:

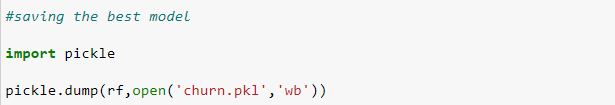


So, the more area under the curve the better the classifier is as compared to the base classifier i.e. a completely random classifier with an AUC score of 0.50.

The AUC score for our classifier is coming out to be 0.8, so our Random Forest model seems to do a good job.

## **Saving the model:**

I am going to use pickle dump to save the best fit model. Pickle module is used for serializing and de-serializing an object structure.



## **Summary:**

We began with the importing of required libraries and loading the dataset. Afterwards, we got an understanding of the dataset by exploratory data analysis, checked about missing data and datatypes. During this process we used seaborn and matplotlib to do the visualizations which helped us in grasping the relation of important features with the target variable. During the data preprocessing part, we imputed the missing data, converted categorical features into numeric ones and converted the datatype where required. Then we proceeded with training 7 different machine learning models, picked one of them (Random Forest) and checked for cross validation. Then we optimized its performance by tuning its hyperparameter values. Afterwards, we came to the evaluation part and checked out its confusion matrix and computed the model’s precision, recall and f1-score. Lastly, we compared our classifier with the base classifier by using ROC Curve and AUC score and saved the model.

There is still a room for improvement. The model’s performance can be further optimized by doing a more extensive feature engineering i.e. comparing and plotting the features against each other and identifying and removing the noisy features. Another way is by doing a better hyperparameter tuning.