# Customer Churn Prediction using ML

## Problem Definition

Customer Churn is when the subscribers of a company stop using their services. This can be due to multiple reasons, with the only possible factor that we can control being the quality of the services we provide. However, there are other factors including the type of services the customer opts for, price, etc which can also be used to predict if a customer would opt to drop out.

Retaining customer is the most important requirement for a company, and working on better ways to improve the retention will definitely boost the company’s revenue and helps the company to build a meaningful relationship with the customer. What might not be so obvious is that customer retention is actually more valuable than customer acquisition and there is a lot of data to back this claim. Furthermore, retaining the existing customers would be cheaper compared to investing in adverts to attract new ones, though both are important.

Looking at the plus sides of creating this model, the company saves money in marketing and its free word-of-mouth advertising that can be leveraged given the customers are satisfied with the quality of the service provided. Repeated purchases/subscriptions mean a continuous inflow of profit for the company and if a customer is really satisfied with the services, there are higher chances for them to opt for the premium tier services offered. All of this, given the feedbacks are taken and their requests accommodated in a feasible manner.

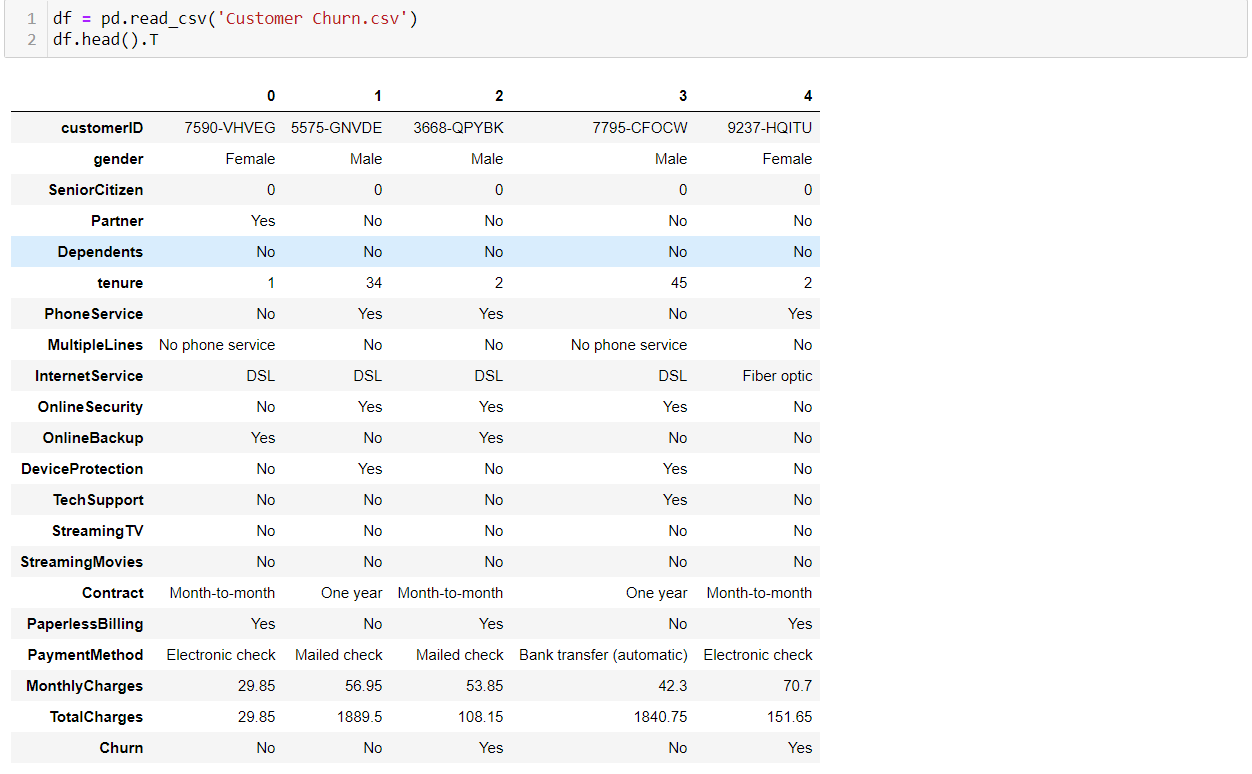
This is an attempt to create a model that can accurately predict / classify if a customer is likely to churn. I’ll be working on customer data from IBM Sample Data Sets with the aim of building and comparing several customer churn prediction models.

## Data Analysis

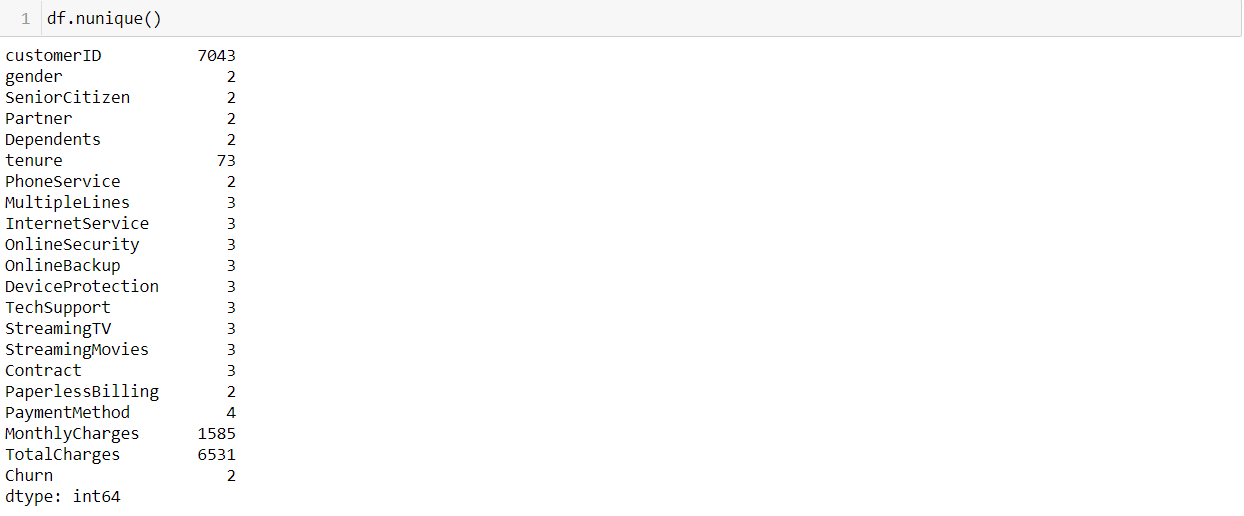
I’ve turned the dataframe sideways in order to give you a view of what I’m working with. To start, we have the column that we need to predict, ‘Churn’, and all the others which we need to figure out what they bring on to the table.

Churn column has two values Y/N, and here, we can outsmart the machine and predict that we will be using a Classification model.

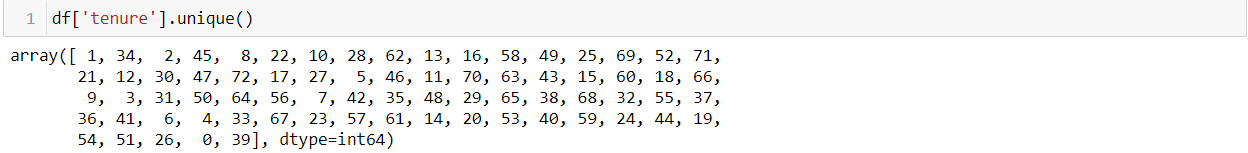
Looking at the other columns we can see a lot which can be considered as categorial values.



Checking the unique values.

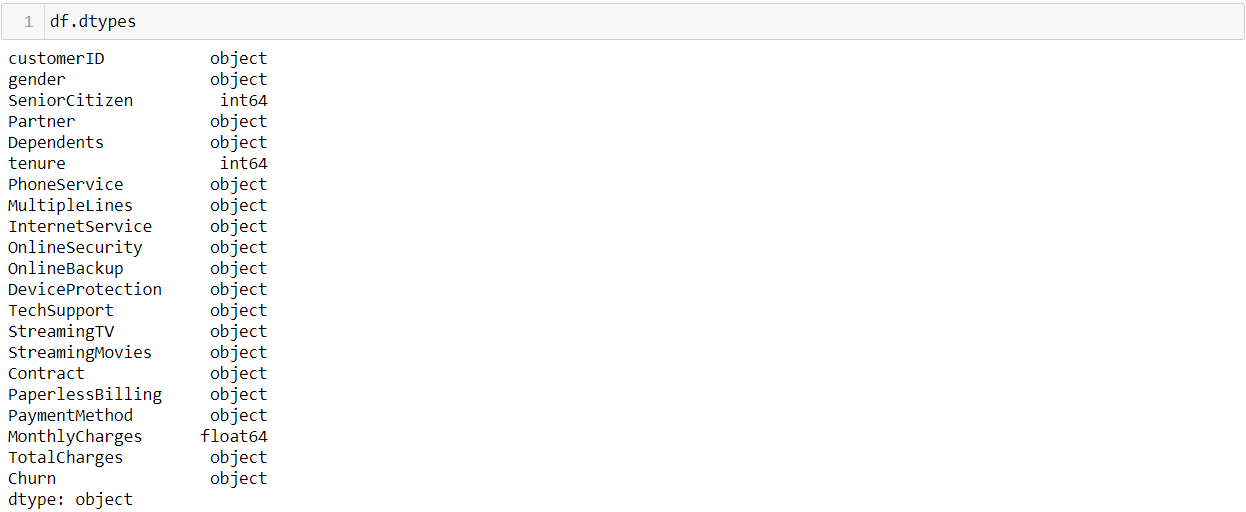


Here we can conclude that most of the columns are categorical with the exceptions being ‘Customer ID’, ‘MonthlyCharges’, and ‘TotalCharges’. I’ll be encoding them later. Can’t be sure about the ‘Tenure’ column though, will check it now.



It can be seen as both categorical and continuous at the same time, and can be left as it is. I was also able to conclude that there are no null values or special characters in this dataset.

However, when checking the dtypes of columns, there was something strange.



‘TotalCharges’ column is supposed to be int/float, but it’s shown as object. This means that there are some undesired values in the column. When checking for blank values, I was able to find that there are 11 rows with missing values.

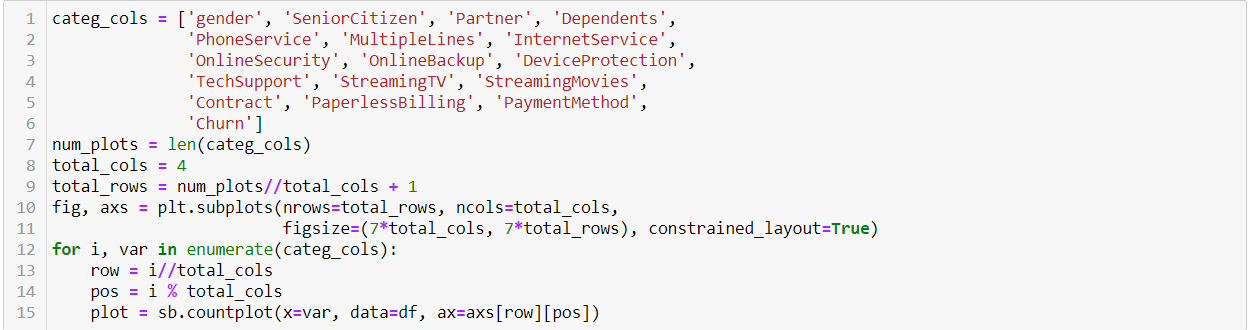
The missing values were turned to NaN values, and the data type was changed to int before imputing. Imputing before changing the datatype causes an error 'can only concatenate str (not "int") to str'.

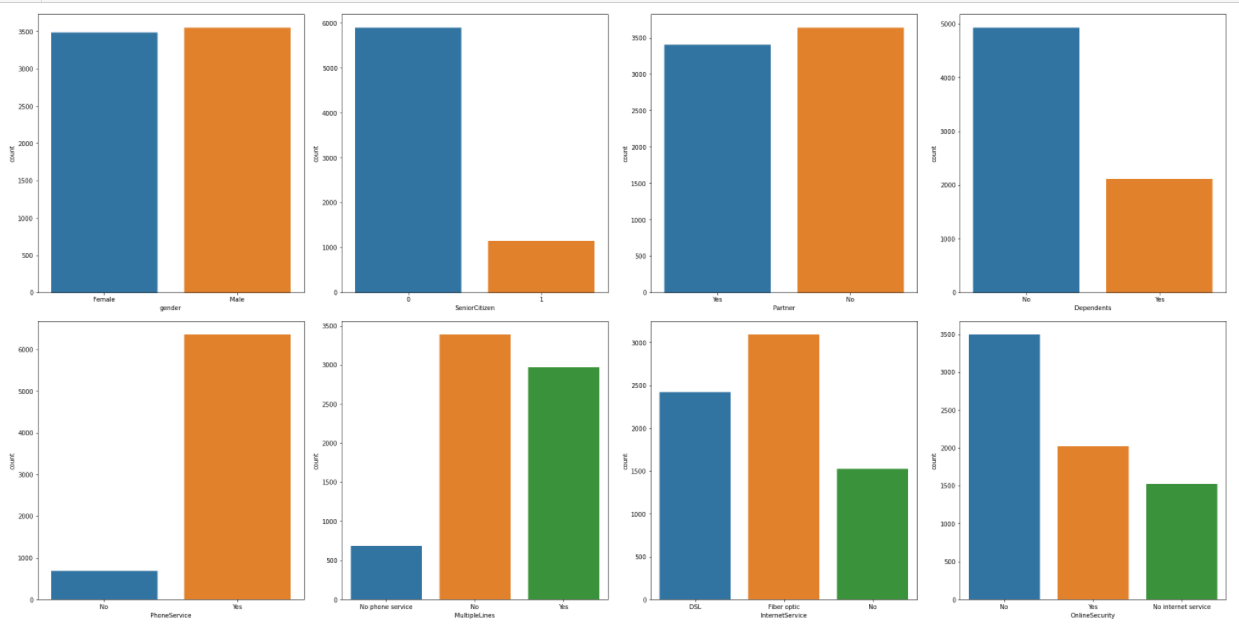


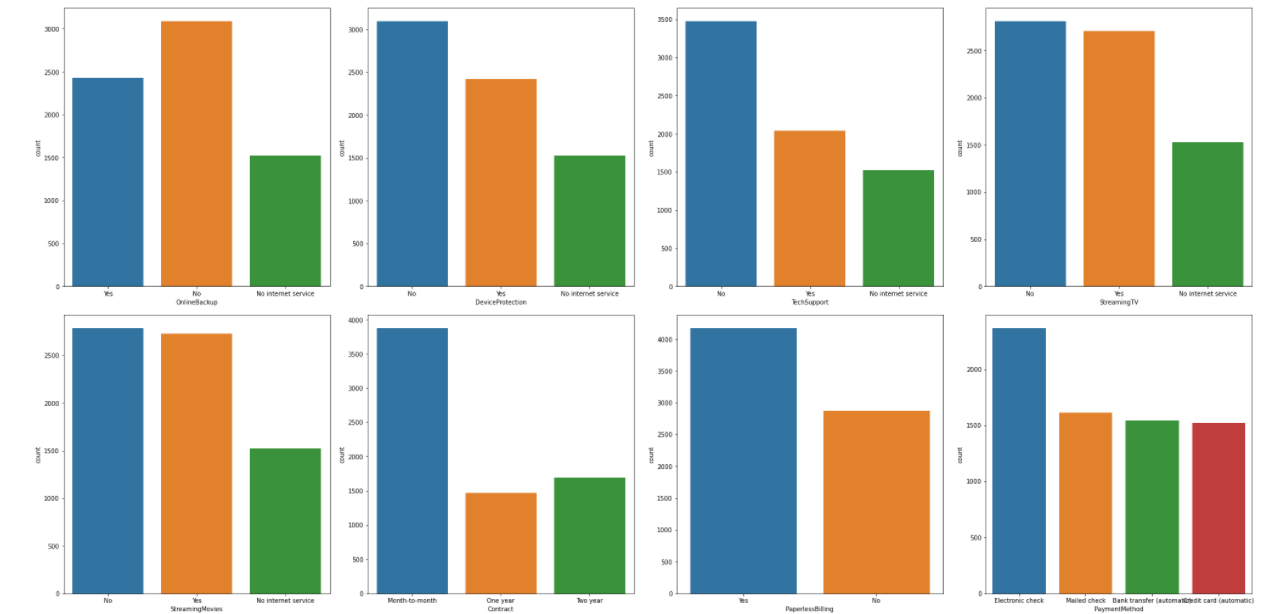


Now the ‘TotalCharges’ column has been changed to float and the missing values imputed.

Proceeding to visualization, I ran a for-loop to visualize all categorical columns in the dataset.





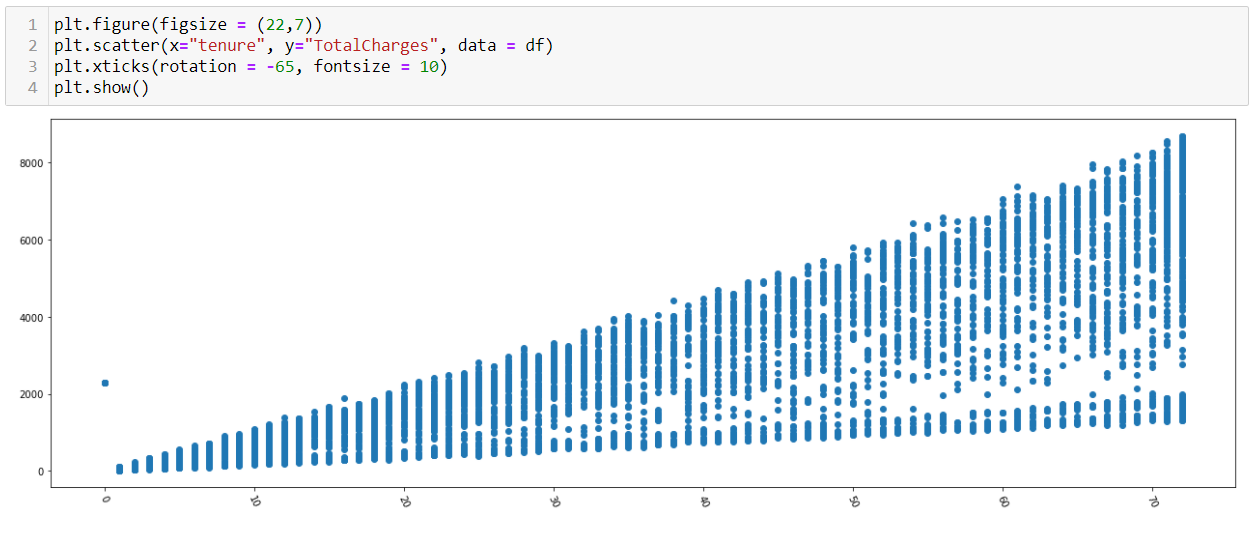


Now that’s a lot of graphs. I’ve noted down the observations so that you don’t have to strain your eyes.

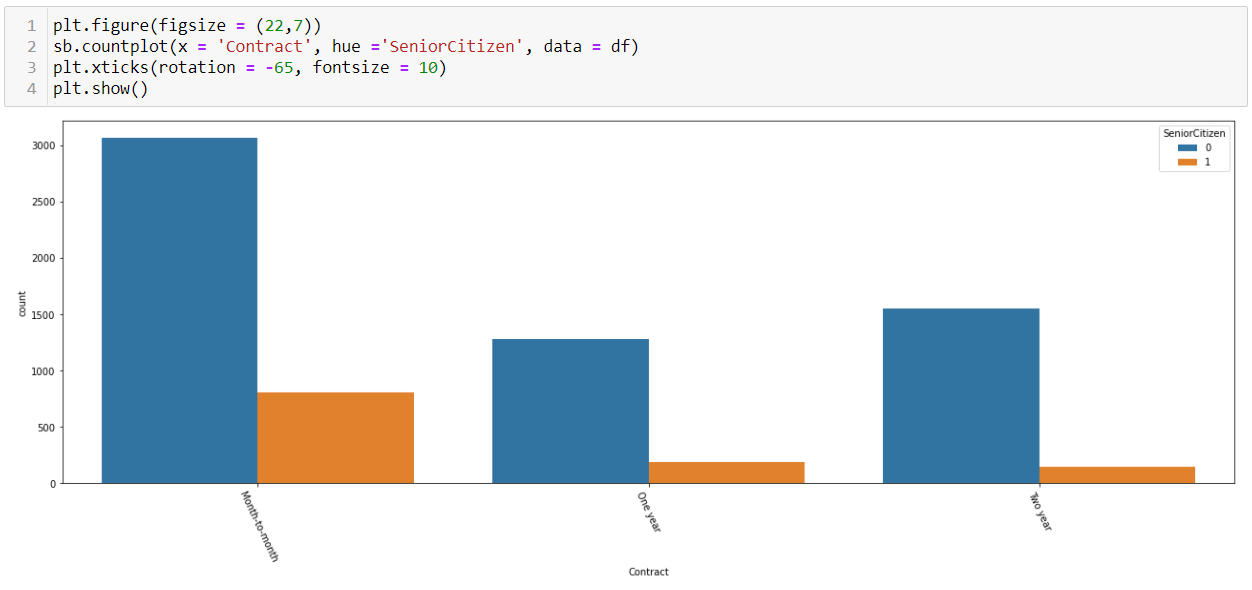
Observation:

* We can see that the dataset is fairly balanced when it comes to the male female ratio.
* We can see that the number of senior citizens in the dataset is fairly low. There are only 1142 SeniorCitizens to 5901 younger people.
* The number of partners & non-partners are fairly equal.
* The number of people with no dependents are higher when compared to people with dependents.
* People who haven't opted for the PhoneService are really low.
* Going with the point above, the people with single lines are higher and the number of people with multiple lines are a tad bit lower.
* 3096 people opted for Fiberoptics, 2421 opted for DSL while 1526 opted for no InternetService.
* 2019 people opted for OnlineSecurity, 3498 didn't opt for it, and 1526 as always has no InternetService.
* 2429 people opted for OnlineBackup while 3088 people didn't.
* 2422 people opted for DeviceProtection while 3095 didn't.
* 2044 people used for TechSupport while 3473 didn't.
* 2707 people used StreamingTv while 2710 didn't.
* 2732 people uses StreamingMovies while 2785 doesn't.
* 3875 people are on month-to-month contract, 1695 on two-year contract and 1473 on yearly renewals.
* 4171 opted for PaperlessBilling while 2872 didn't.
* 2365 opted for Electronic check, 1612 opted for Mailed check, 1544 opted for Bank transfer (automatic), 1522 opted for Credit card (automatic)
* 5174 didn't Churn while 1869 turned to other operators.

Plotting the tenurity vs total charges, I’ve found a strange relation between them.



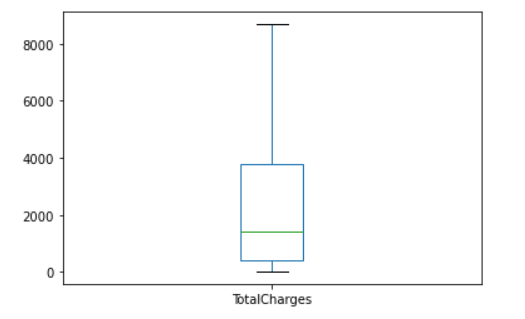
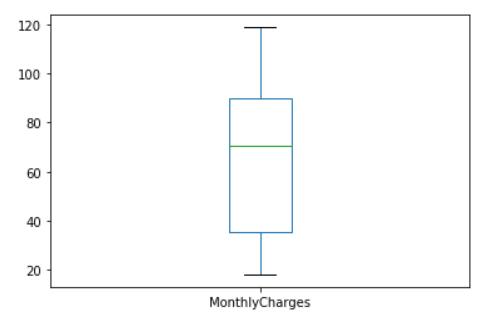
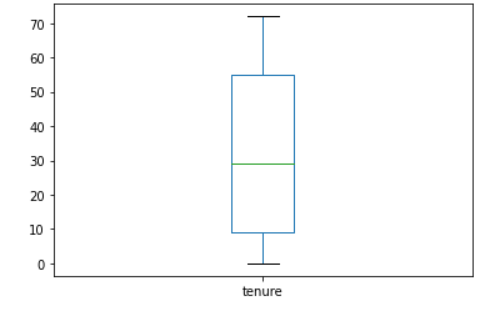
The Total charges/the services utilized by a person tend to increase the longer they stay with a company.



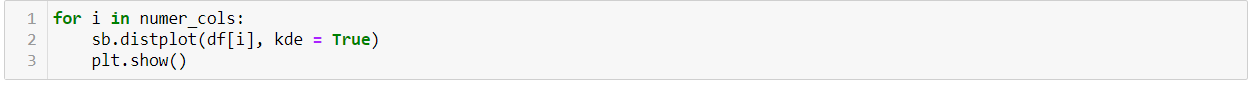
Looking at the contract periods, we can sese that the highest number of all people tend to use month-month connection while comparatively, higher number of non-senior citizen tend to use one & two-year connections.

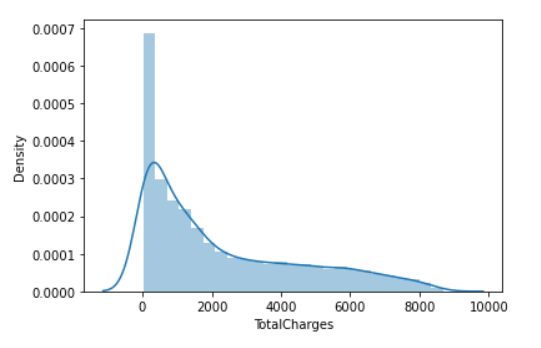
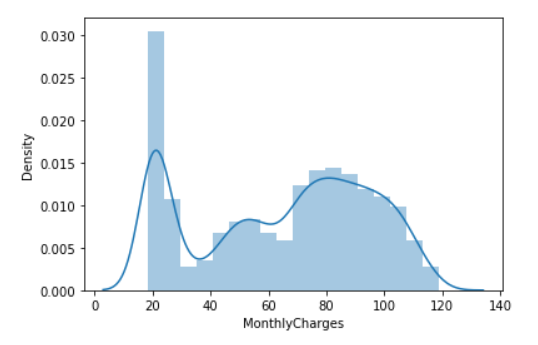
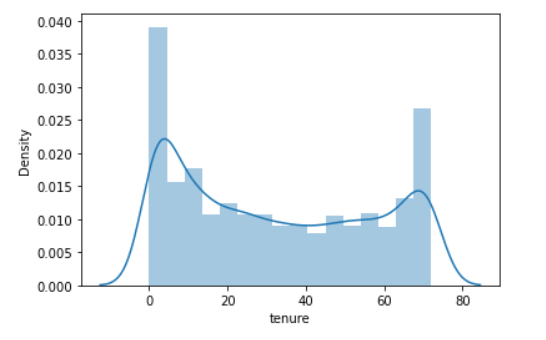
**Checking for outliers**: I was unable to find any outliers in any of the continuous columns.

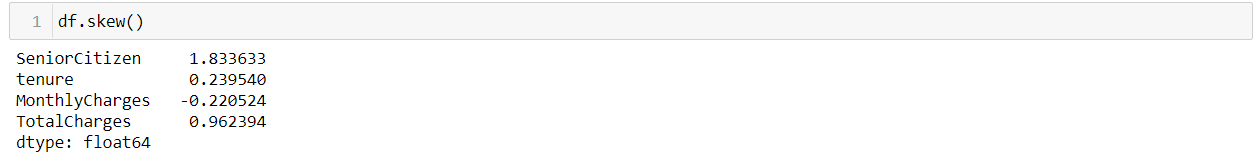




**Checking for Skewness**:



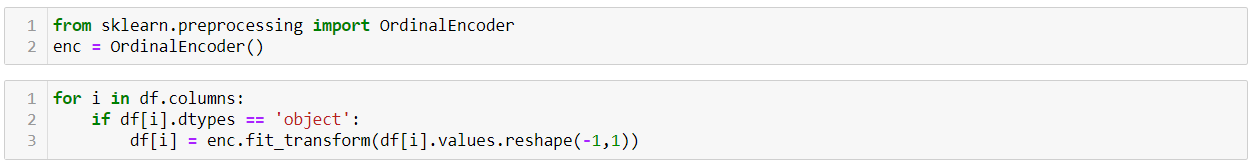




* We can see skewness in TotalCharges. We will remove this before modelling.
* Since all the other columns except the ones in numer\_cols are categorical, we can ignore those.

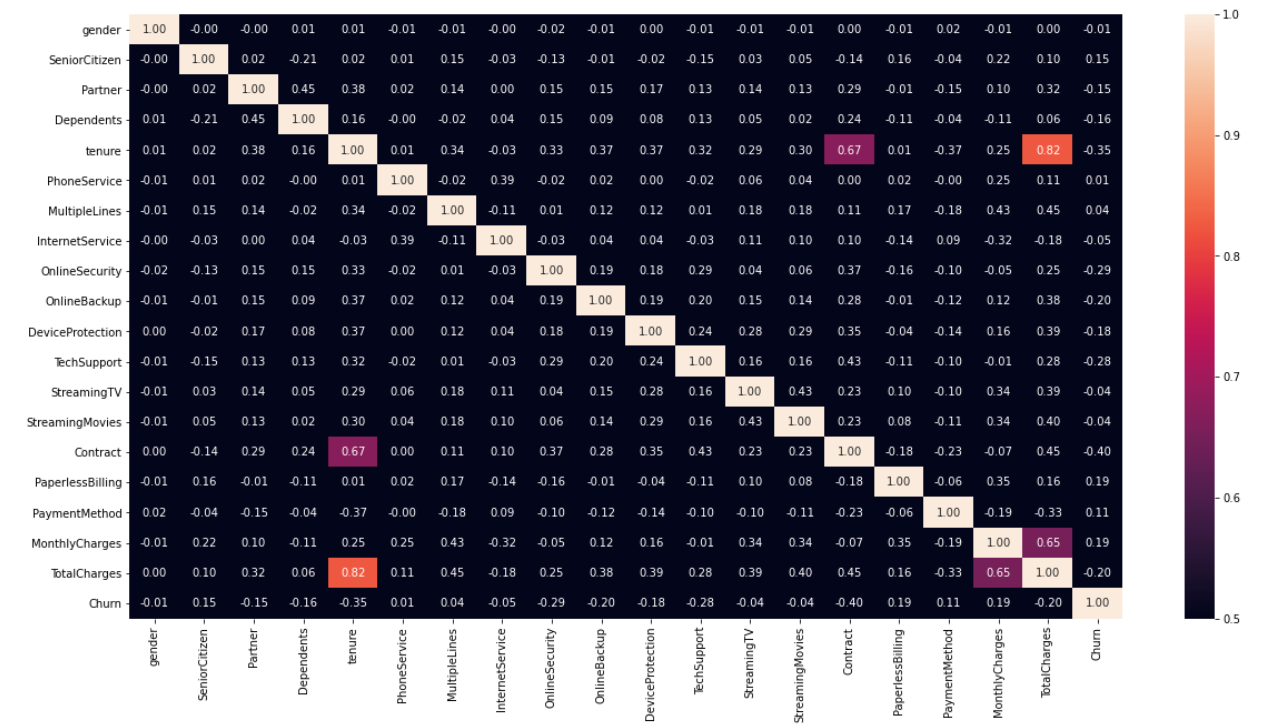
**Encoding**: I’ve used OrdinalEncoder to encode the categorical columns into ones and zeros. An ordinal encoding involves mapping each unique label to an integer value. This type of encoding is really only appropriate if there is a known relationship between the categories. This relationship does exist for some of the variables in our dataset, and ideally, this should be harnessed when preparing the data.

In this case, we will ignore any possible existing ordinal relationship and assume all variables are categorical. It can still be helpful to use an ordinal encoding, at least as a point of reference with other encoding schemes. We can use the OrdinalEncoder to encode each variable to integers. This is a flexible class and does allow the order of the categories to be specified as arguments if any such order is known.



**Describing the dataset**: I was able to see high difference in values for mean and max in the column ‘TotalCharges’. Customer ID is unique for everyone, so we can remove the whole column.

**Correlation**: While plotting the heatmap for correlation, I was unable to find any high negative correlation between the columns. On the other hand, there were only two points where positive correlation was present.



Observation:

* We can see that there is high correlation between Tenurity and TotalCharges.
* There is a high correlation between Contract and tenurity.
* High correlation between MonthlyCharges and TotalCharges.

**Outliers**: While checking for the outliers using the ZScore, I was able to find all the outliers present were in the PhoneService column. Since this is a categorical value, we can ignore it. Hence, we can conclude that there are no outliers in the dataset.

**Scaling**: After splitting the X and Y variables, I’ve scaled the values to remove When the range of values are very distinct in each column, we need to scale them to the common level. The values are brought to common level and then we can apply further machine learning algorithm to the input data. The values in the continuous columns are quite high when compared to the smaller encoded values in the categorical columns. Hence, this is an important step to get a good model.

StandardScaler removes the mean and scales each feature/variable to unit variance. This operation is performed feature-wise in an independent way. StandardScaler can be influenced by outliers (if they exist in the dataset) since it involves the estimation of the empirical mean and standard deviation of each feature. Since there are no outliers, the scaling should work perfectly.

## EDA Concluding Remarks

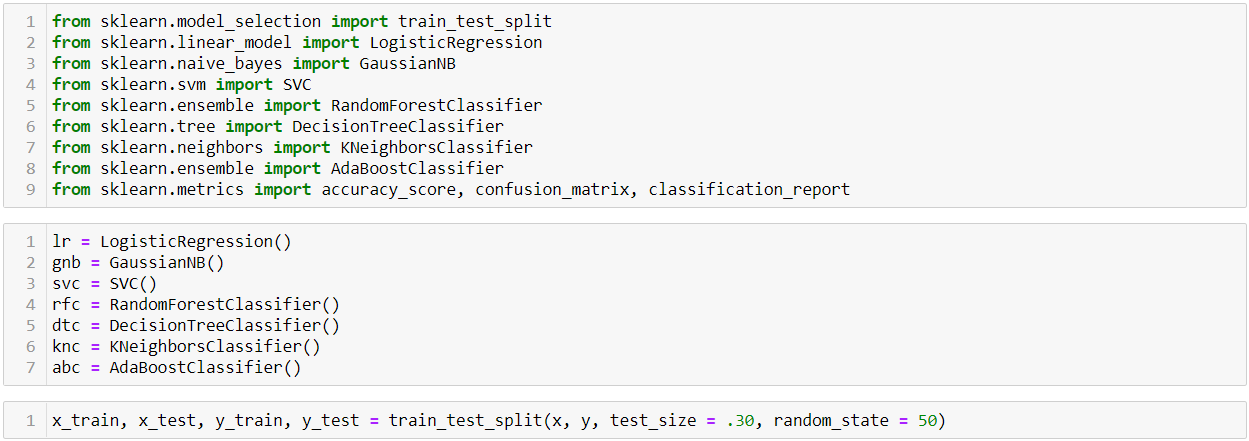
* Removed the ‘customerID’ column
* Checked the datatypes, and found a fault with the ‘TotalCharges’ column.
* Found blank values in ‘TotalCharges’, changed it to NaN, and imputed it with mean.
* Extrapolated details from all categorical columns.
* Checked for outliers.
* Checked for skewness.
* Checked the summary and correlation, and noted down the findings.

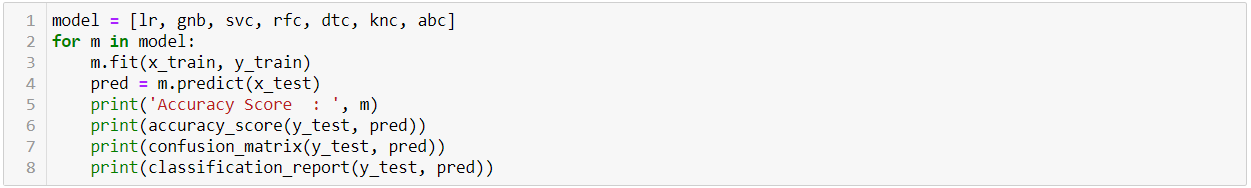
## Pre-processing Pipeline

* Encoded the object datatypes using OrdinalEncoder.
* Checked for outliers, but didn’t remove it since it was in a categorical column.
* Scaled the dataset using StandardScaler.

## Building Machine Learning Models

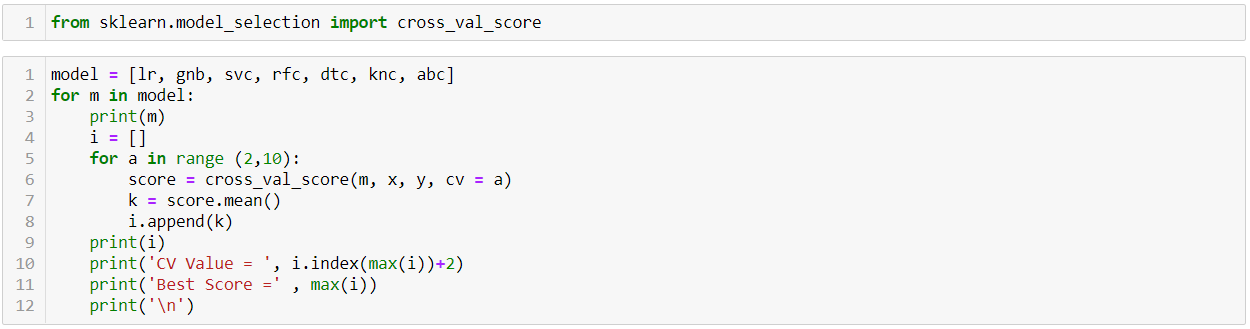
Once the X & Y variables are split, I’ve imported all the models and setup a for-loop to find the models with the best accuracy.





Running the for-loop above, I’ve with me LogisticRegression, SVC, and RandomForestClassifier giving good accuracy scores. All three has values really close to each other with the differences in decimals.

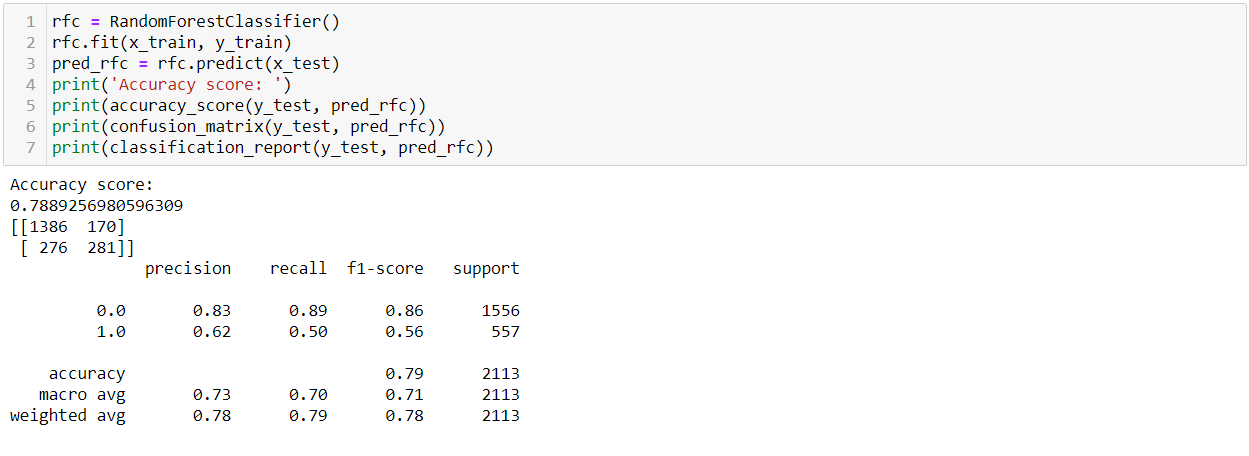
**CrossValidation**: Given below is the for-loop that I’ve ran to check the over/under-fitting for all the models.



This will give me an output which shows the best split for each model. Once this is done, I’ll note down the R2 Score and the CV score, and find the model with the least difference. I was able to find the model with the least difference between Cross Validation Score and the initial Accuracy score to be RandomForestClassifier.

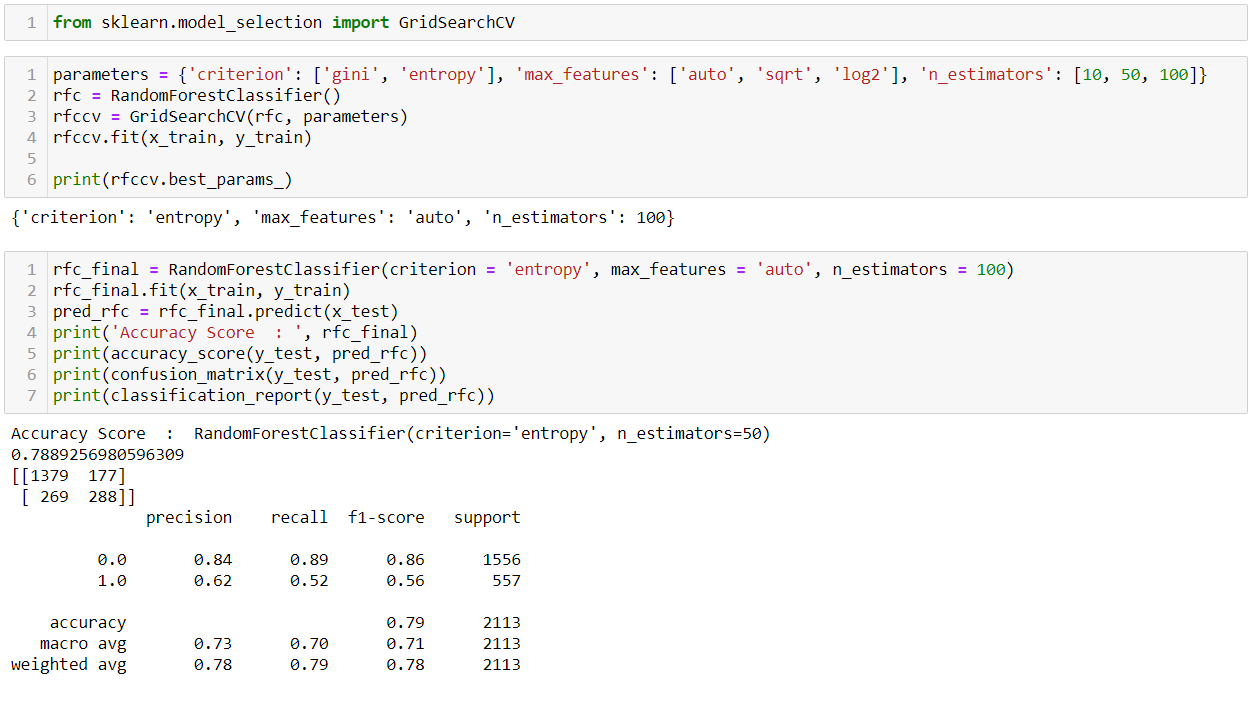
The best value for RandomForesetClassifier was at CV = 8 and it gave me a score of 79%, which isn’t much different when compared to the initial accuracy score.

I’ve re-run the best model once again to set proper instances which will help further down the line while plotting the AUC-ROC Curve.



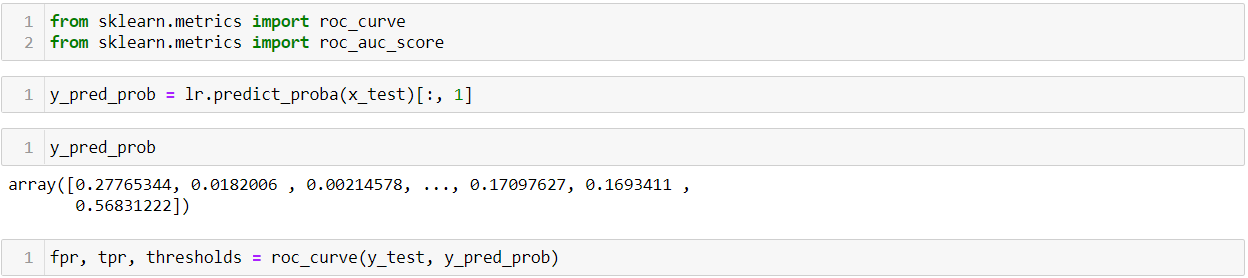
**GridSearchCV**: GridSearchCV tries all the combinations of the values passed in the dictionary (which I’ve set manually) and evaluates the model for each combination using the Cross-Validation method. Hence after using this function, we get accuracy/loss for every combination of hyperparameters and we can choose the one with the best performance.

It helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, you can select the best parameters from the listed hyperparameters.

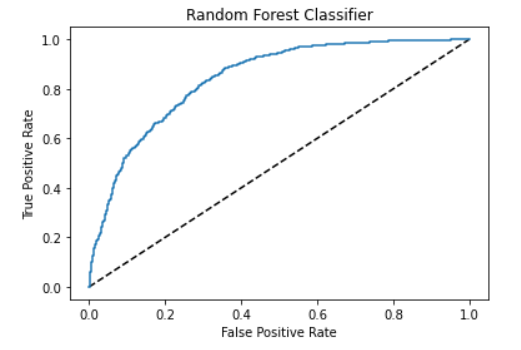


With this, we’re able to find that the best criterion for the RFC is with ‘entropy’, max\_features set to ‘auto’, and ‘n\_estimators’ value set to 100. This is giving us a model with an accuracy of almost 79%.

**AUC-ROC Curve**: This is 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.



Plotting the AUC-ROC Curve, I got a graph as shown below.



AUC - ROC curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represent the degree or measure of separability. It tells how much the model is capable of distinguishing between classes.

Checking the AUC score of the plot above, I was above to find that the model is giving me a score of 72% which is lower than the initial, cross validation, as well as GridsearchCV scores.

**Saving the model**: I’ve used the Joblib module to build the ML model. It provides utilities for saving and loading Python objects that make use of NumPy data structures, efficiently.



## Concluding Remarks

Churn rate is a health indicator for subscription-based companies. The ability to identify customers that aren’t happy with provided solutions allows businesses to learn about product or pricing plan weak points, operation issues, as well as customer preferences and expectations to proactively reduce reasons for churn.

To conclude, the mode I’ve created can predict the churn in the months to come with a reasonable accuracy of 72% to 79%