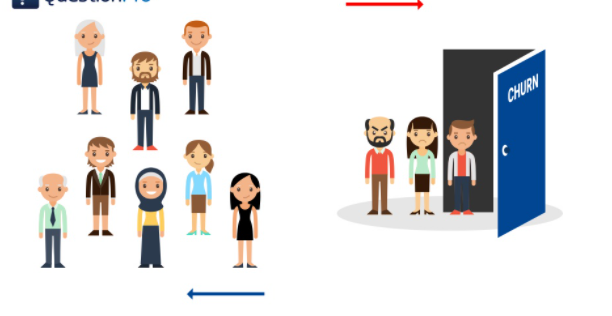
Predicting the Customer churn

**Problem Statement**:

**Customer churn** is when a company’s customers stop doing business with that company and with increase in number of businesses, awareness a very common phenomenon these days.



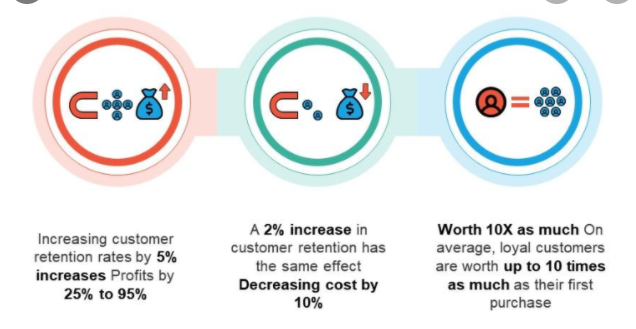


Businesses are very keen on measuring churn because keeping an existing customer is far less expensive than acquiring a new customer. New business involves working leads through a sales funnel, using marketing and sales budgets to gain additional customers. Existing customers will often have a higher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



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 prioritize focused marketing efforts on that subset of their customer base.

It has been observed that the customer retention has proved to have significant benefits in company’s profit



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

The approach to understand the churn would be to understand the answers to the below question:

* Do we want to keep this customer?
* Why are they leaving?
* What can we do?
* What will the effect of our intervention be?
* How long will the intervention effect persist for?

In this document , we will try to analyse the data , clean it and prepare for model building and choosing the best model

**Data Analysis**

* Importing Libraries & Algorithms

*#Importing libraries*

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

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

**from** **math** **import** \*

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

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

**import** **warnings**

warnings.filterwarnings("ignore")

%matplotlib inline

*# Algorithms*

**from** **sklearn** **import** linear\_model

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

**from** **sklearn.ensemble** **import** RandomForestClassifier

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

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

**from** **sklearn.tree** **import** DecisionTreeClassifier

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

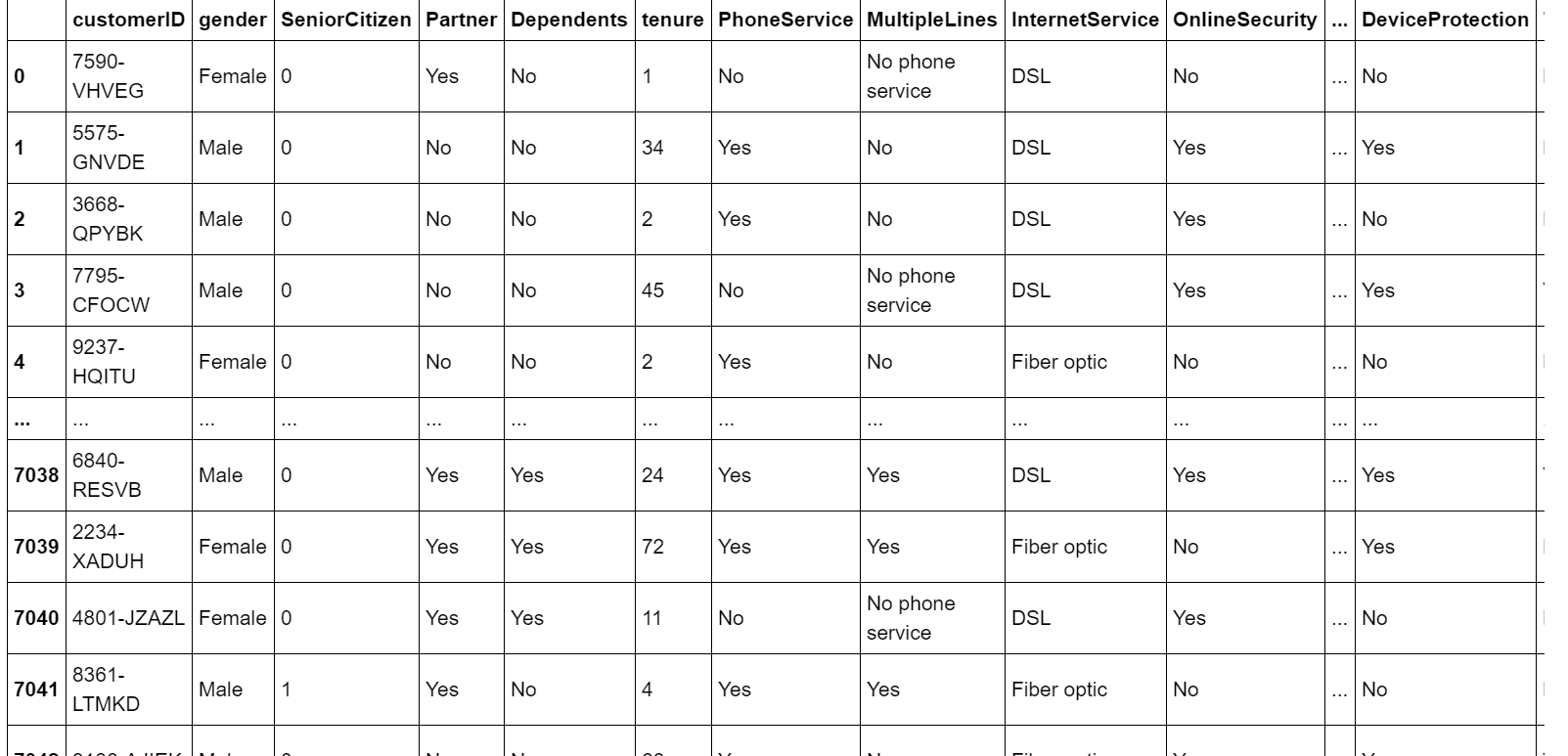
**from** **sklearn.svm** **import** SVC, LinearSVC

**from** **sklearn.naive\_bayes** **import** GaussianNB

* Getting the data

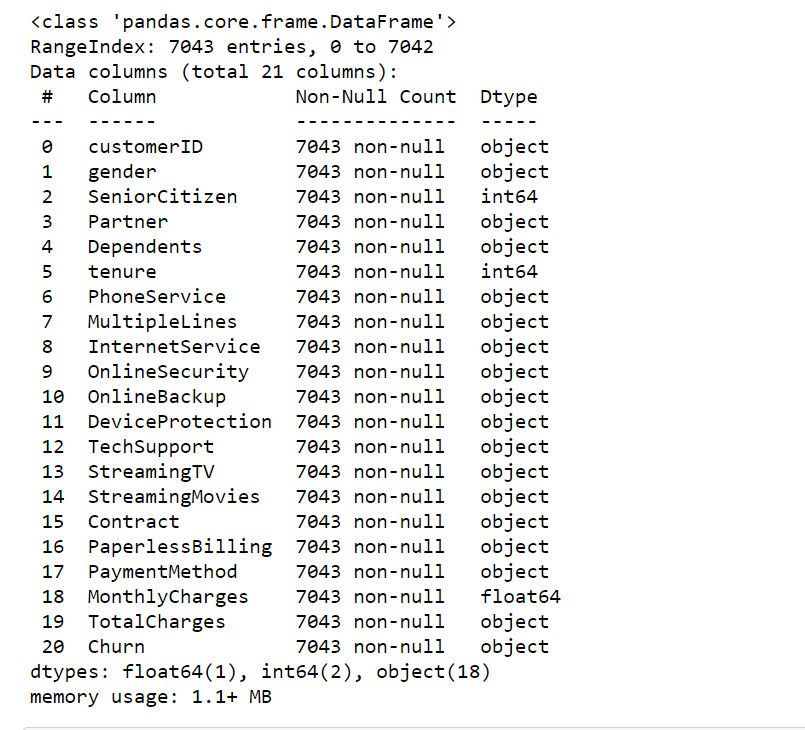
df=pd.read\_csv("Telecom\_customer\_churn.csv")

df

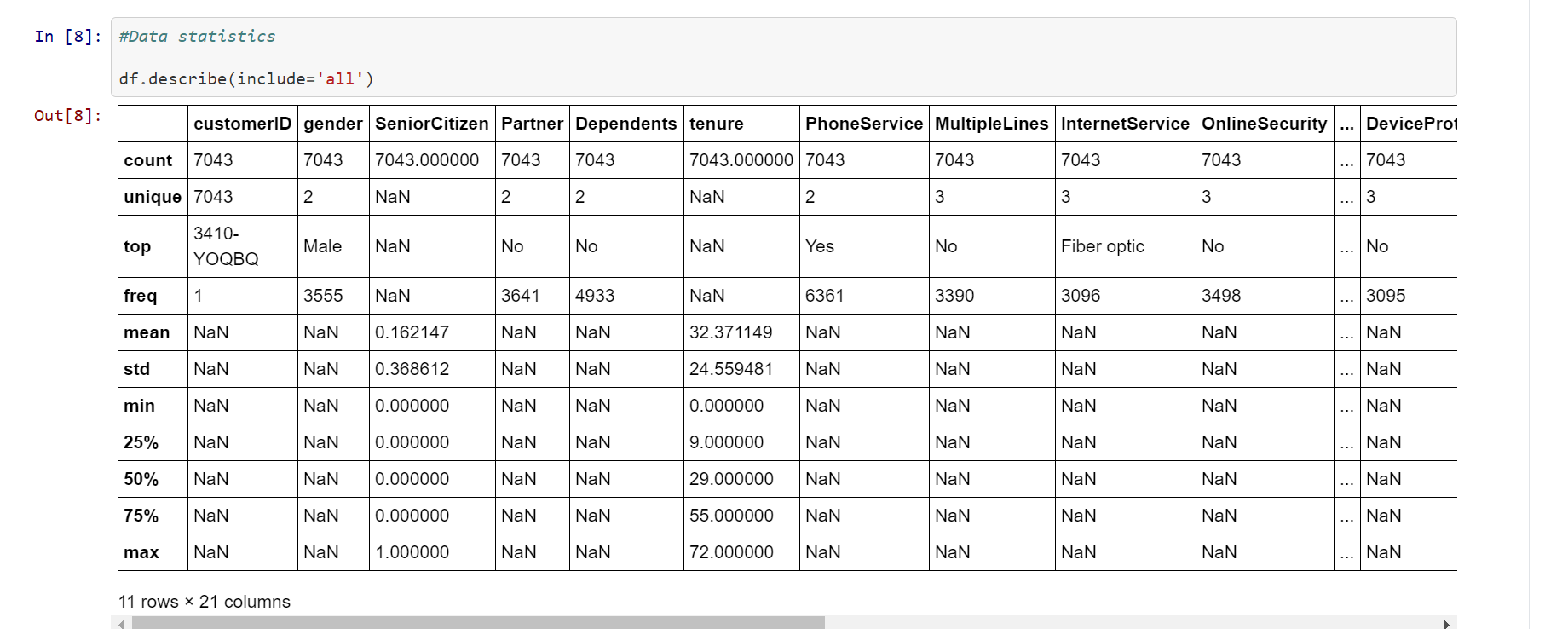


* To understand the data type of dataset

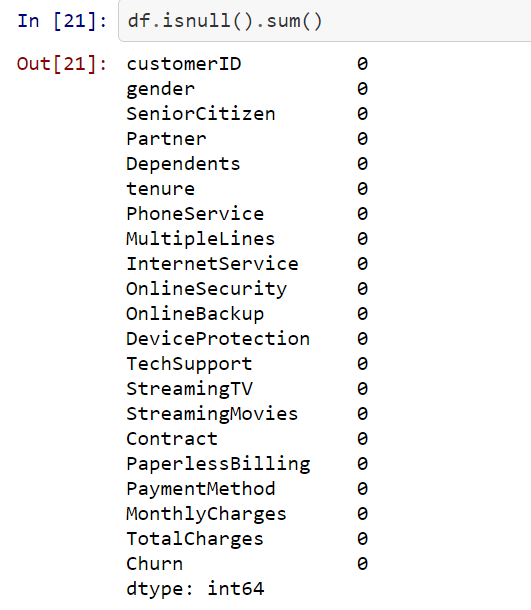
df.info()



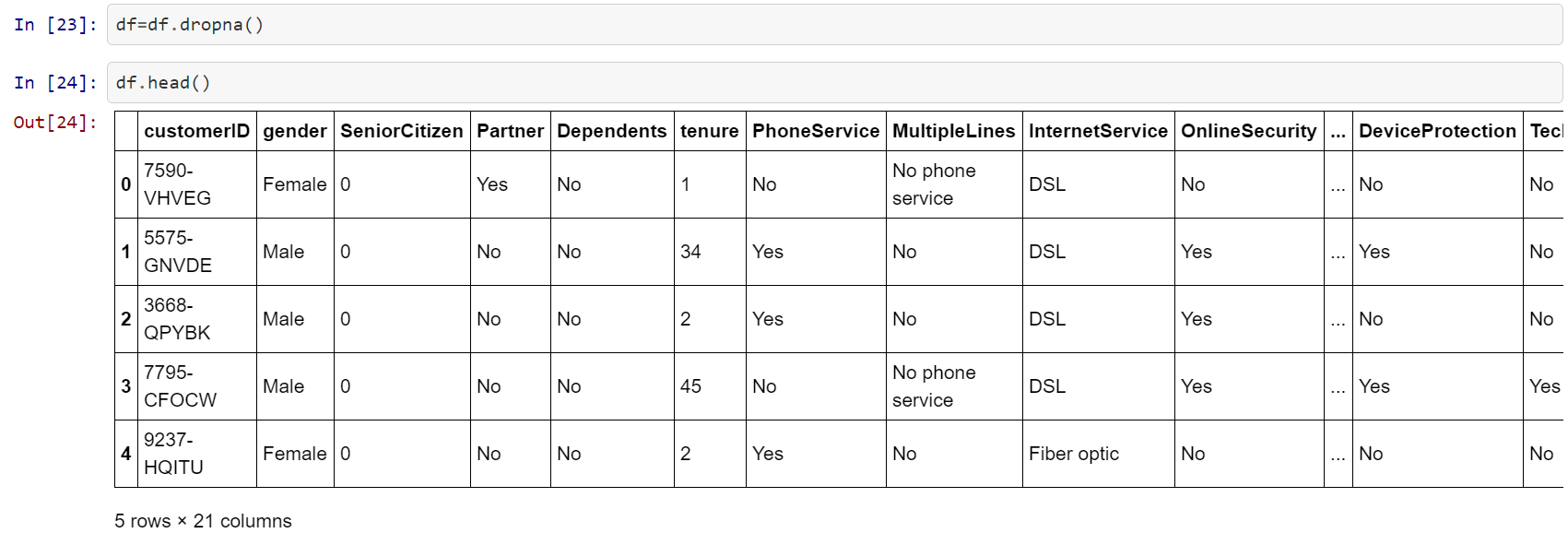
* Let’s check on the data Statistics



* As we can see lot of missing / NAN values lets check on the missing values



We can see no Missing value however we could see a lot of Nan values.Dropping the Nan values



Nan values have been removed .Now lets look at the unique values in each column .

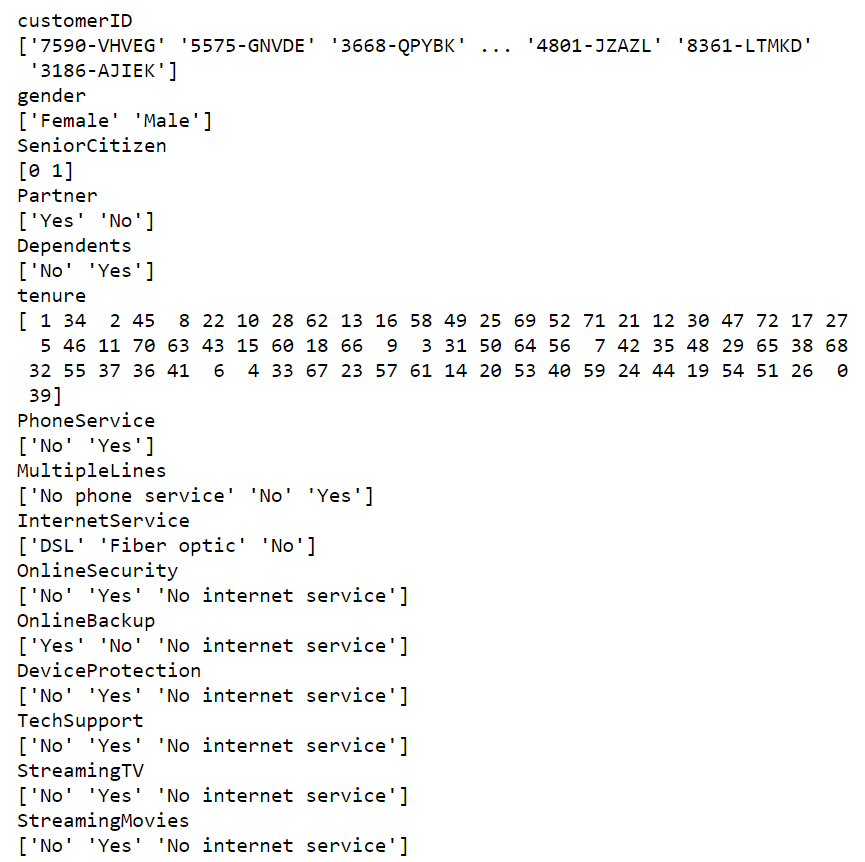
**Data Exploration &Visualization,Preprocessing**

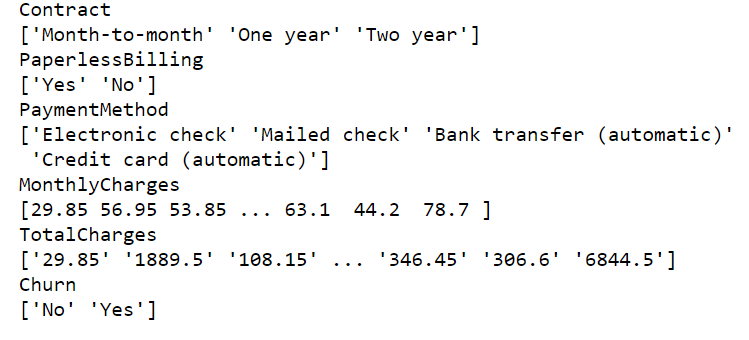
* To view the unique values for each column

**for** item **in** df.columns:

print(item)

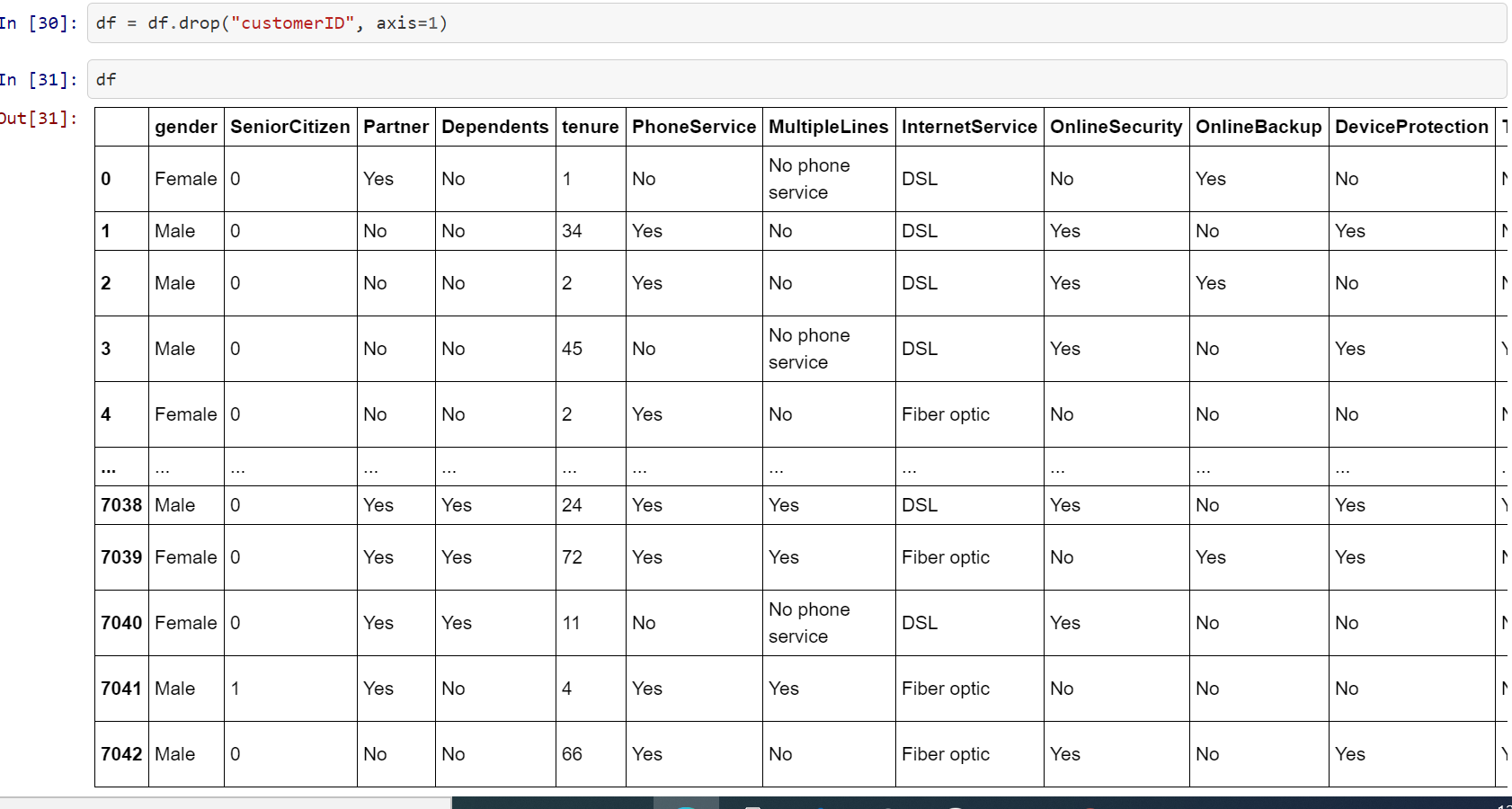
print (df[item].unique())



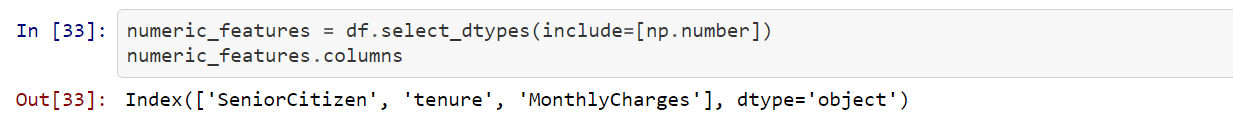


* Dropping the unwanted columns

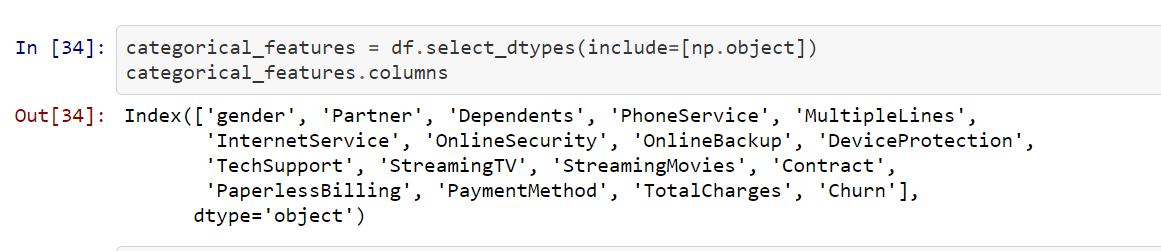
Customer id will not be affecting the customer churn



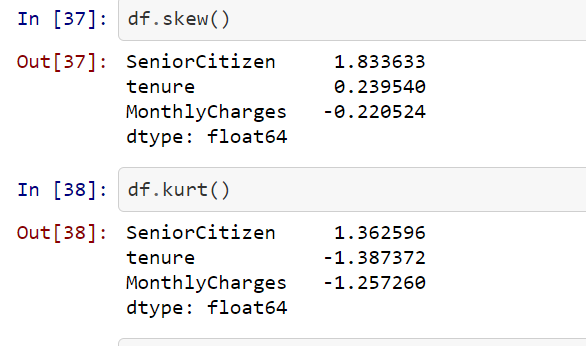
* Let’s check for the numerical and categorical features of the dataset
* Let’s check for numerical dataset



* Let’s check for categorical features

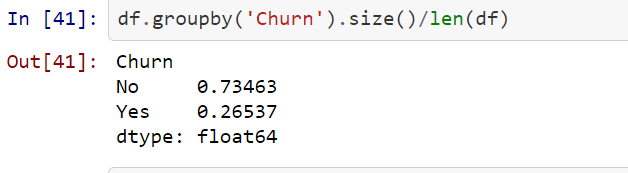


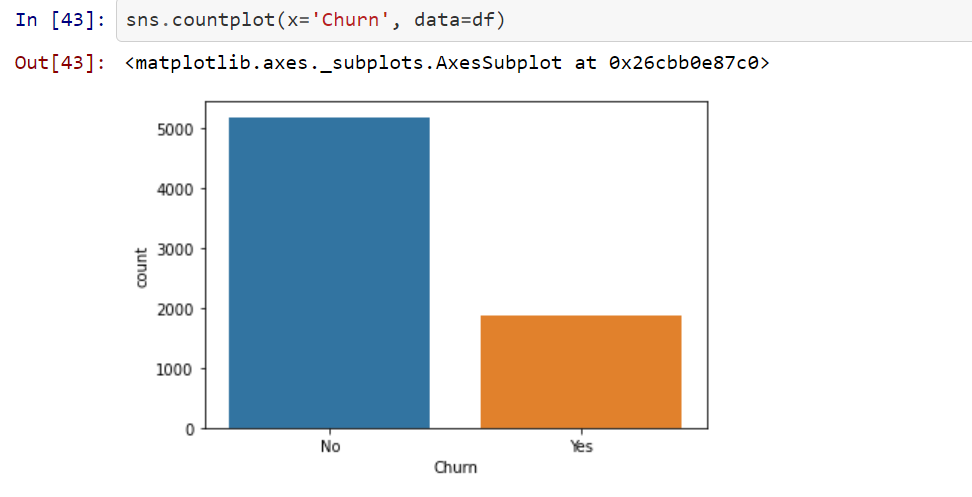
* To figure out the skewed data



**Data Visualization**

* Let’s do some data visualization
* Let’s explore the target variable to understand the percentage of churners

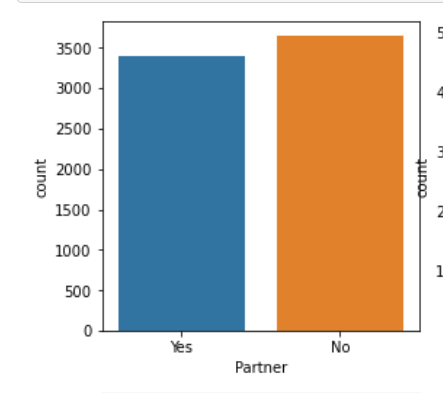




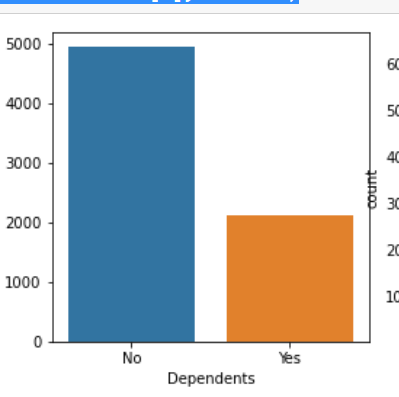
* Univariate Analysis for all categorical columns

plt.figure(figsize=(18, 18)) **for** k **in** range(1, len(categorical\_features.columns)): plt.subplot(4, 4, k) sns.countplot(x=categorical\_features.columns[k], data=df)

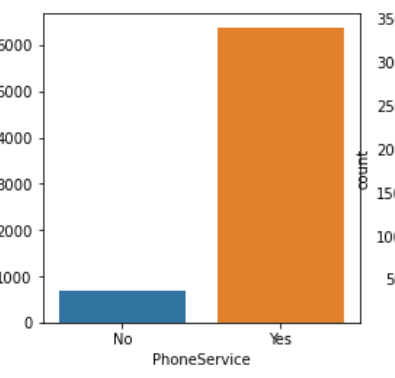
* **Partner**



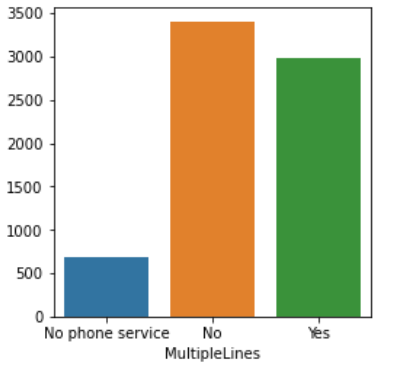
* Dependents



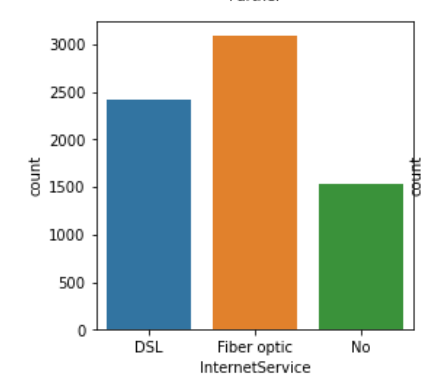
* Phone Service



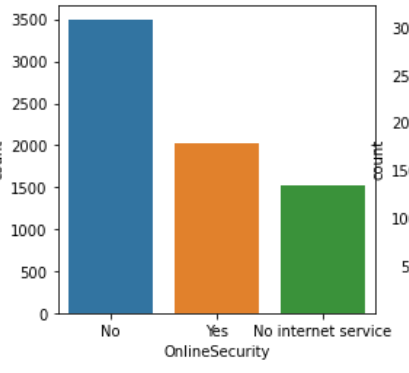
* Multiple Lines



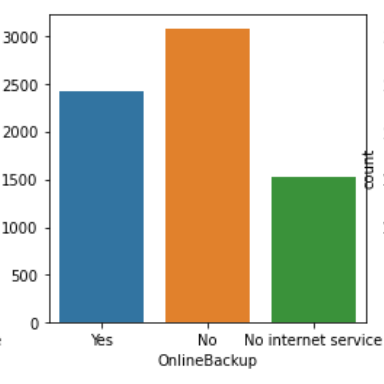
* Fibre Optic Service



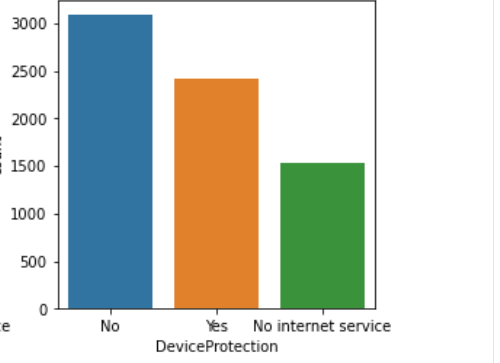
* Online Security



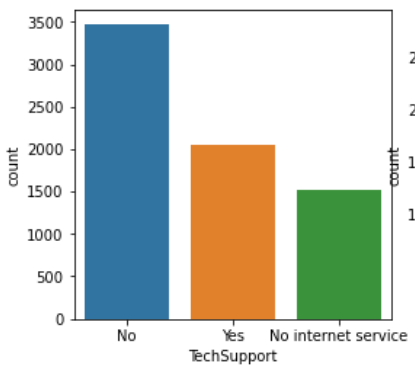
* Online backup



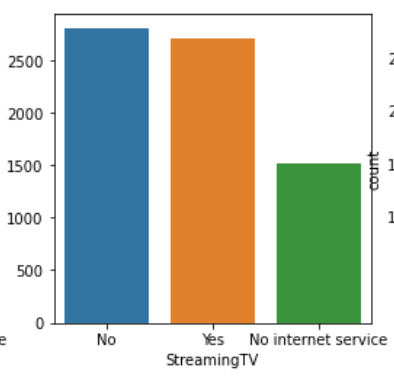
* Device Protection



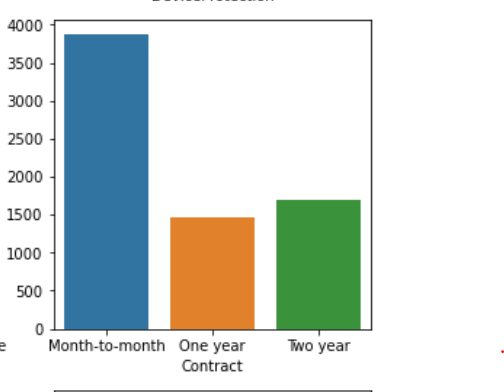
* Tech Support



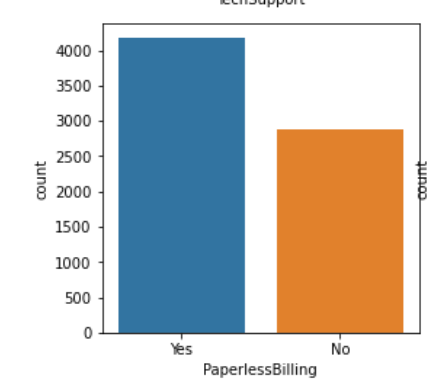
* Streaming TV



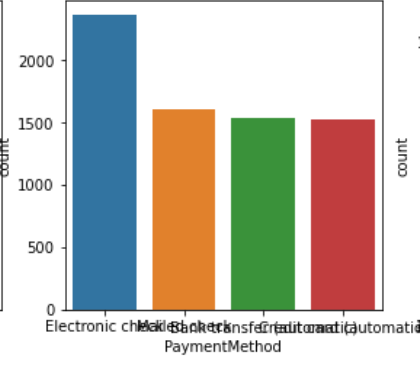
* Contract



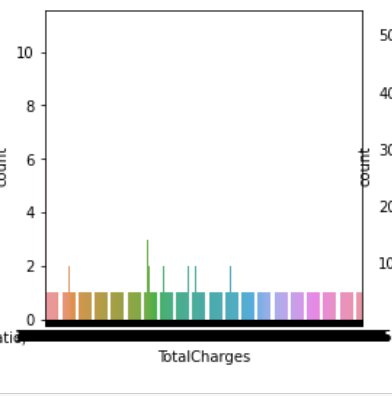
* Paperless Billing



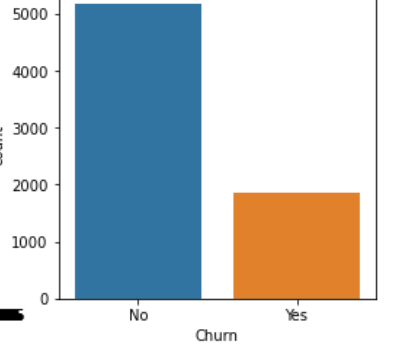
* Payment method



* Total Charges



* Churn



* **Bivariate Analysis**
* Let’s analyse the features against the target variable

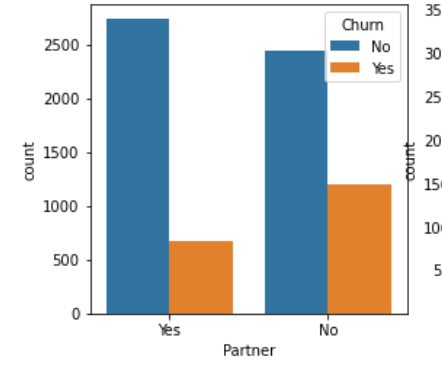
plt.figure(figsize=(18, 18))

**for** k **in** range(1, len(categorical\_features.columns)):

plt.subplot(4, 4, k)

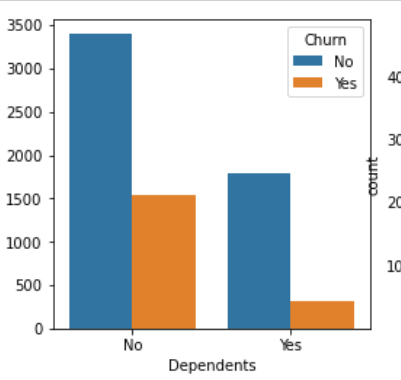
sns.countplot(x=categorical\_features.columns[k], data=df, hue='Churn')

* **Partner against Churn**



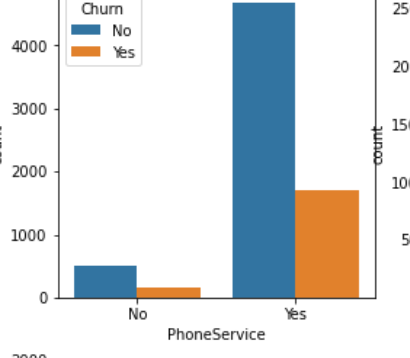
Customers with partner have a lower rate of churn than the customers without partner

* **Dependents Vs Churn**



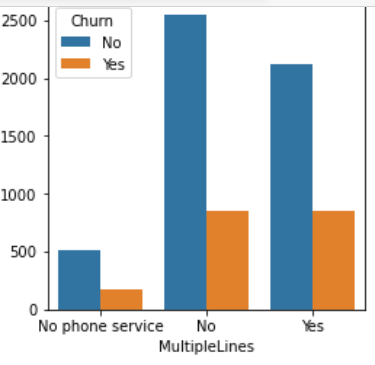
Customers with dependents have lower probability of churning

* **Phone Service Vs Churn**



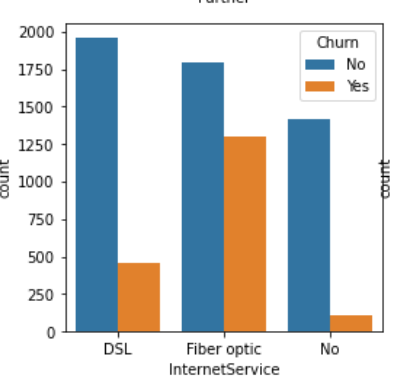
Customers with phone service are more likely to churn

* **Multiple lines vs Churn**



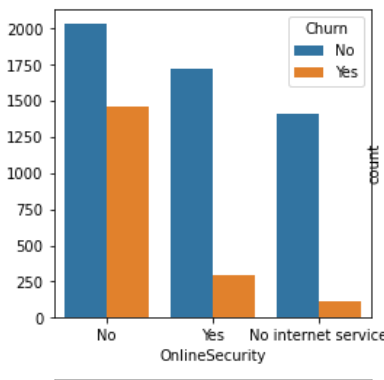
Customers with no phone service are the least likely to churn followed by No multiple lines

* **Internet Service vs Churn**



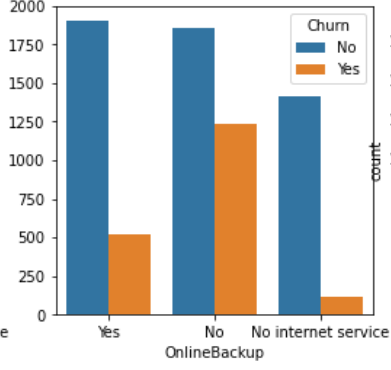
Customers with fibre optic service are the most likely to churn.

* **Online Security Vs Churn**



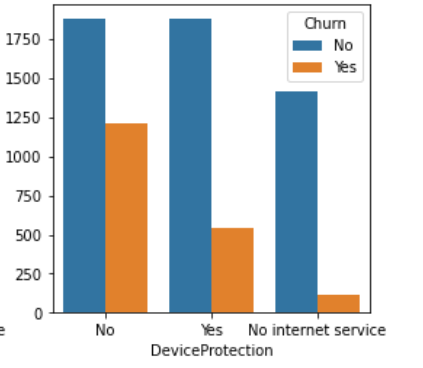
Customers with no online security are the most likely to churn.

* **Online Backup vs Churn**



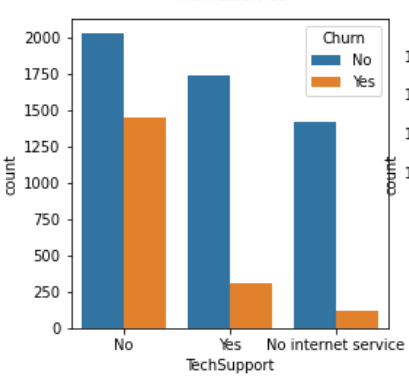
Customers with no OnlineBackup are likely to churn.

* **Device Protection vs Churn**



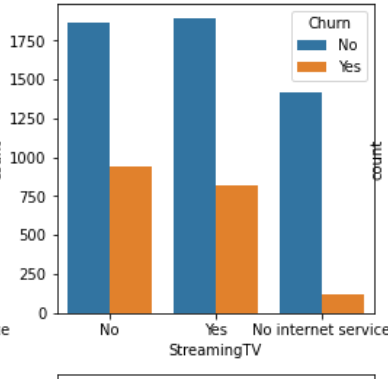
Customers who have no device protection are likely to churn

* **Tech Support vs Churn**



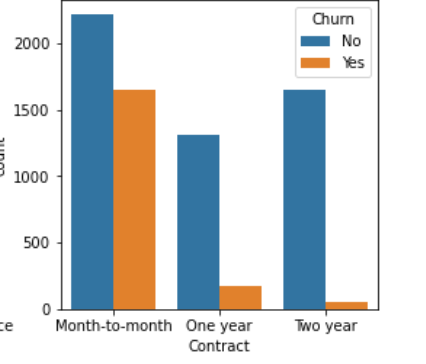
Customers with no tech support are very likely to churn.

* **Streaming TV vs Churn**



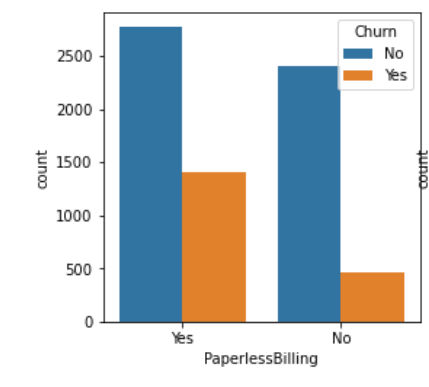
Customers with no Streaming TV are more likely to churn

* **Contract vs Churn**



Customers with short duration contract are more likely to churn.

* **Paperless Billing vs Churn**

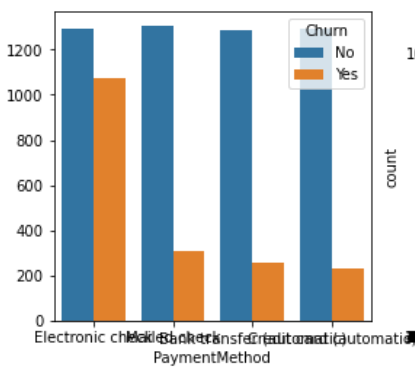


Customers who have paperbilling are more likely to churn

* **Payment method vs Churn**

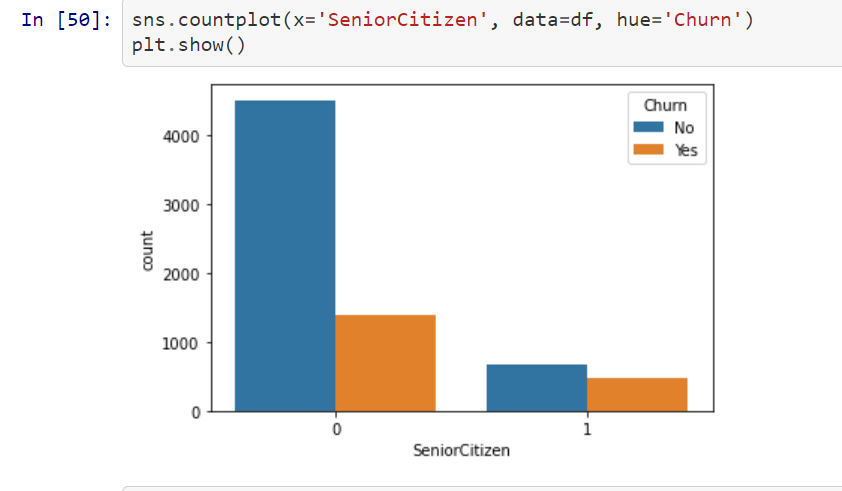
Customers who paid via Electronic method are likely to churn

Customers who paid via Electronic method are likely to churn



Customers who paid via Electronic method are likely to churn

* SeniorCitizen Vs Churn

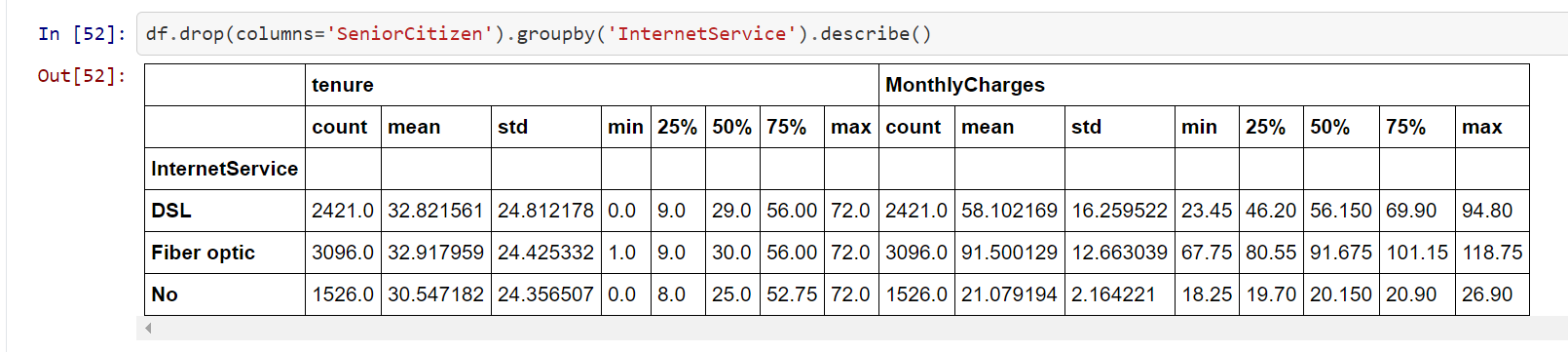


Seniorcitizens are less likely to churn

* **Observations**

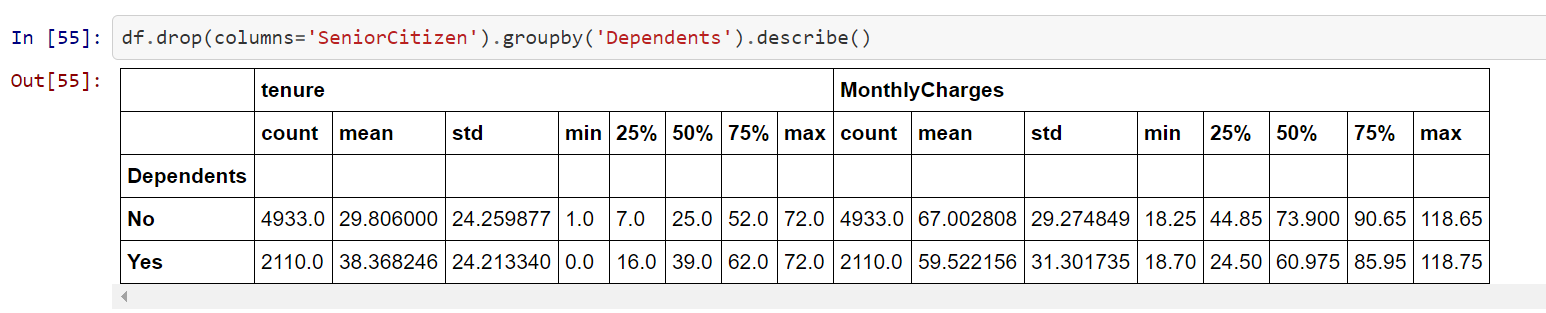
Customer churning happened for the below conditions:

* Among those who have optical fiber,
* Those who have no technical support,
* No device protection probably due to a lack of quality.
* Those who have a month-to-month contract are more likely to churn
* Those who have no streaming movies are likely to churn
* Let’s analyze the understand some of these features that we saw above in details

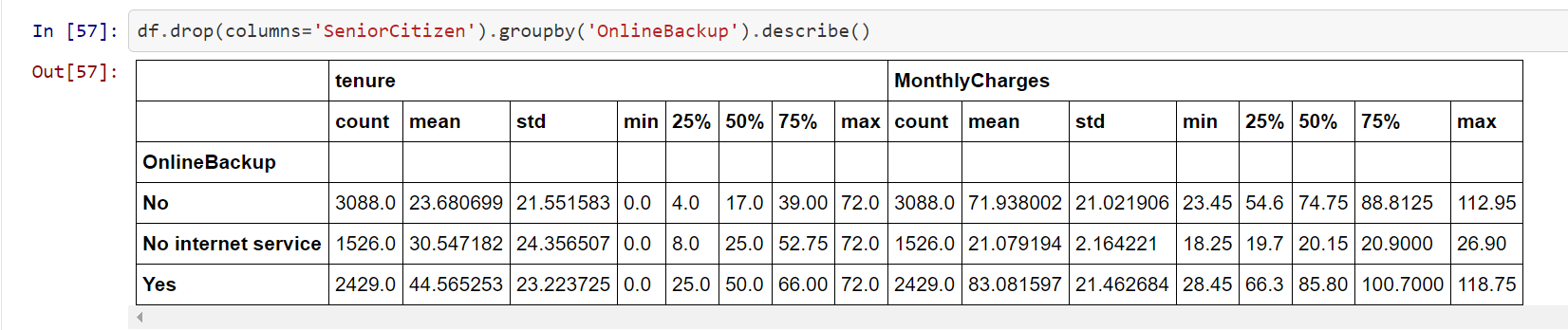


We have seen previously that there is a very high churn rate for those who have a fiber optic service, but we can observe in these statistics that they are also those who bring the most money to the company on average. They are definitely customers that must be kept.

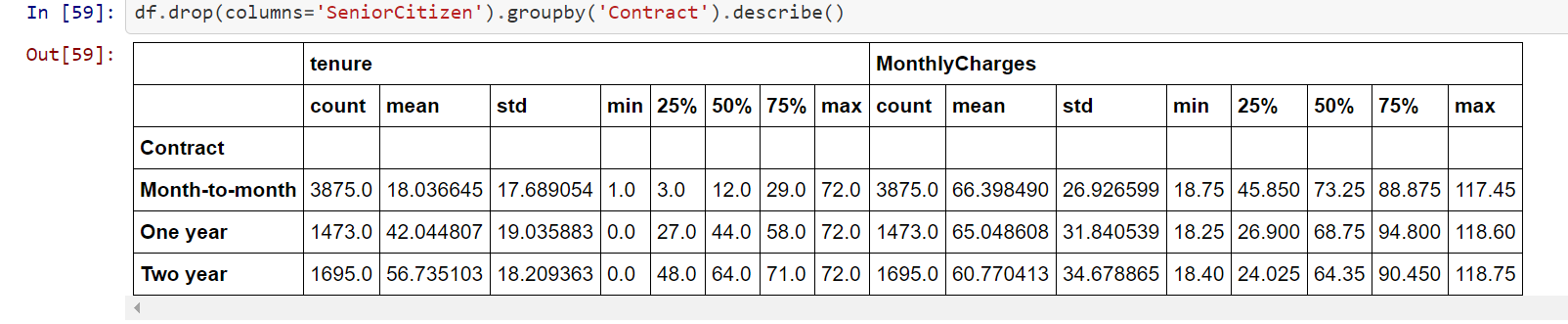
df.drop(columns='SeniorCitizen').groupby('Partner').describe()



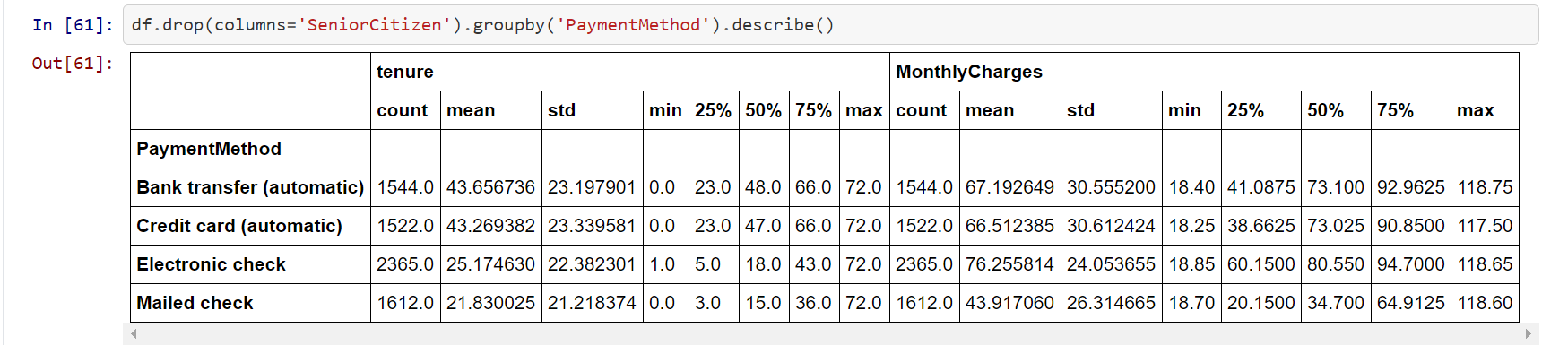
Customers with higher Revenue must be retained.



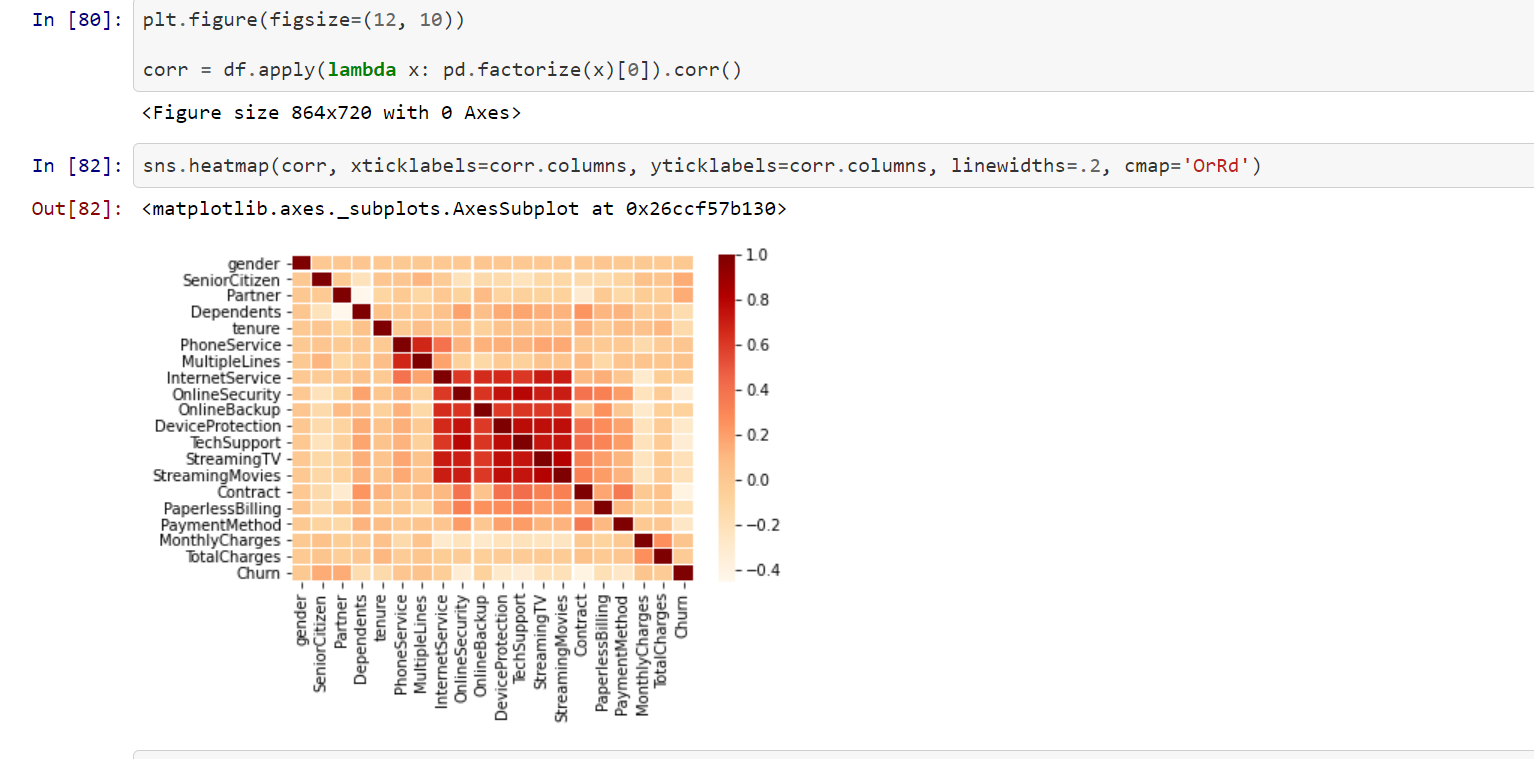
Customers with online backup yield more profit to the company and more likely to use hence should be retained



Customers with long term contract are more loyal and hence customers should be advised for long term contracts.



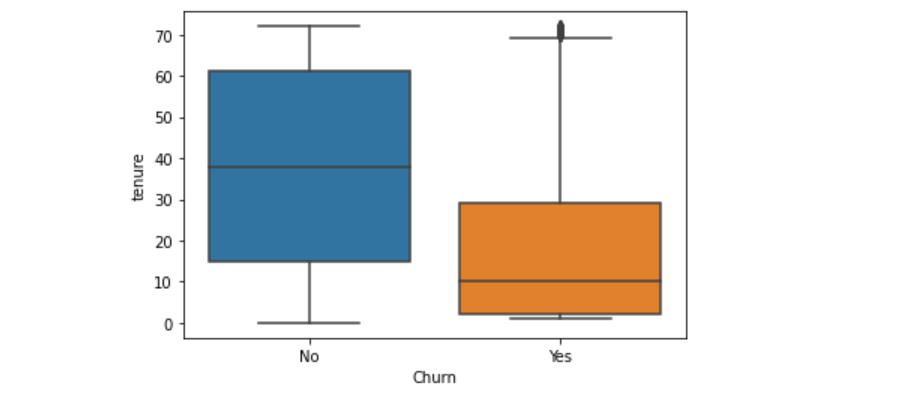
* Correlation Matrix
* Let’s plot the correlation matrix, the darker a box is, the more features are correlated



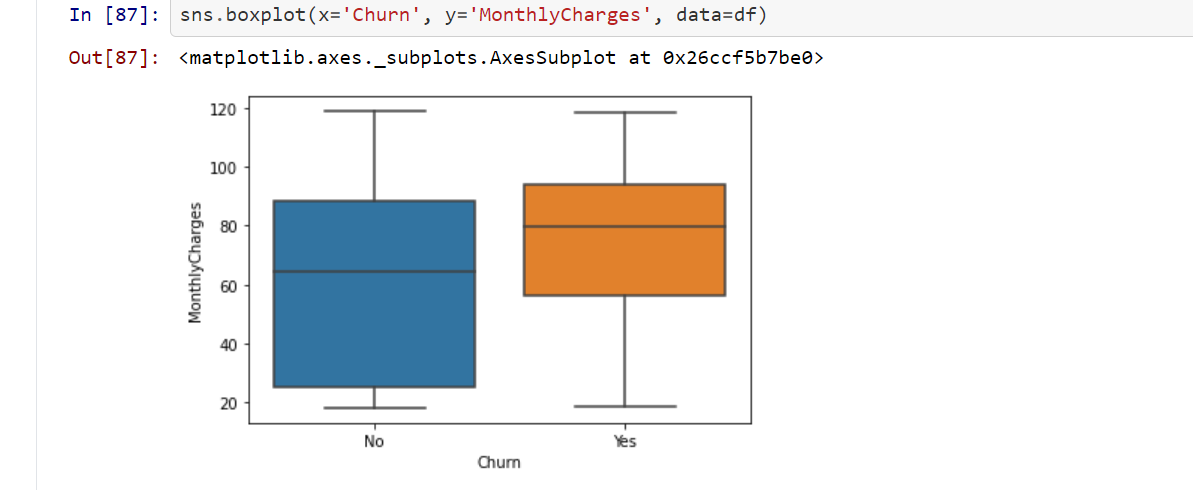
**Observations from the correlation matrix**

* Internet service, Online security, Online Backup, DeviceProtection, Tech Support and streamingTV,Streaming movies,contract are highly correlated features
* Total charges and customer ID are also very correlated, maybe the ID is chosen according to high-potential customers
* The most correlated to churn : Senior, Partner, Multiple lines, online backup, Monthly charges

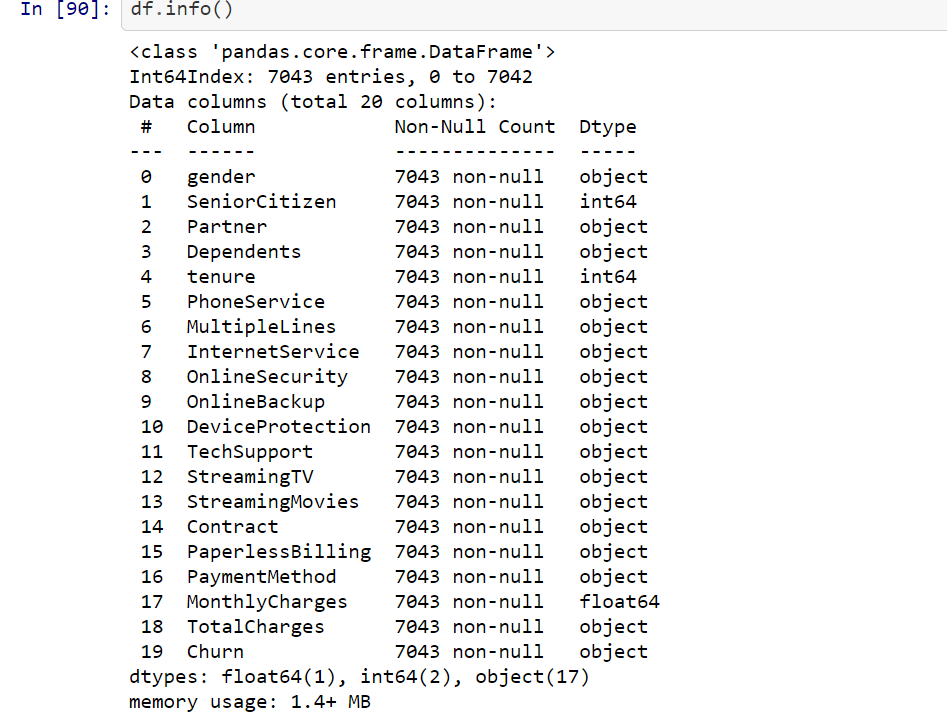
**Visualise and analyse the numerical features against churn**



* Lesser tenure has a higher number of churn



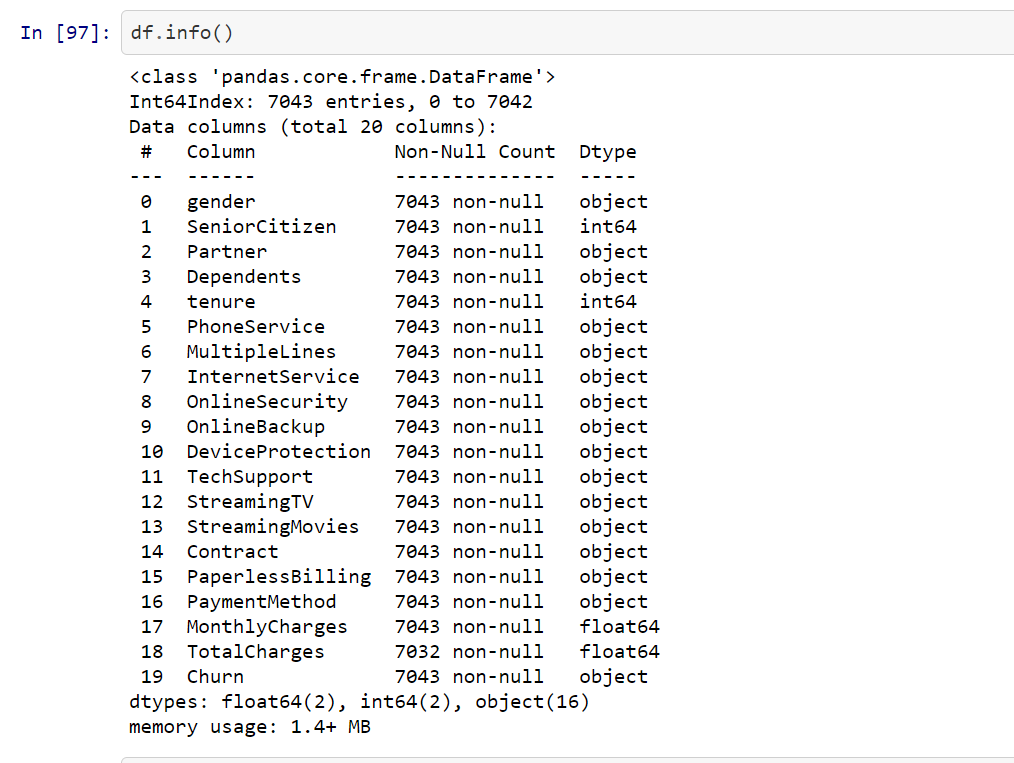
* The ones with higher monthly charges are likely to churn more

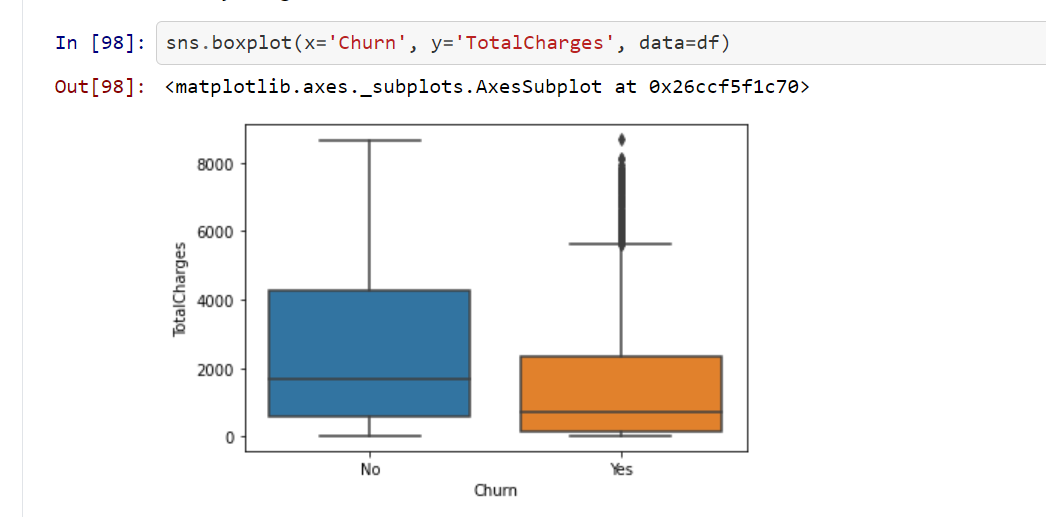


* To process the totalcharges column which has numerical data but is of type object. As we cannot calculate anything with string values, we have to convert these values intro numeric ones.

df=df.replace(to\_replace=" ",value=np.nan)

df.TotalCharges=pd.to\_numeric(df.TotalCharges)





* To convert the categorical features into numerical. As we cannot calculate anything with string values, we have to convert these values intro numeric ones.

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

le = LabelEncoder()

df = df.apply(LabelEncoder().fit\_transform)

* Splitting the data. First our model needs to be trained, second our model needs to be tested. Therefore it is best to have two different dataset. As for now we only have one, it is very common to split the data accordingly. X is the data with the independent variables, Y is the data with the dependent variable. The test size variable determines in which ratio the data will be split. It is quite common to do this in a 70 Training / 30 Test ratio.

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

X = df.drop("Churn", axis=1)

y = df["Churn"]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)

**Model Building:**Predicting the churn is a classification problem and hence we will be using the below classifiers for model building.

* **Random Forest** : The **random forest** is a **classification algorithm** consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated **forest** of trees whose prediction by committee is more accurate than that of any individual tree.

*#Random Forest:*

random\_forest = RandomForestClassifier(n\_estimators=100)

random\_forest.fit(X\_train, y\_train)

Y\_prediction = random\_forest.predict(X\_test)

random\_forest.score(X\_train, y\_train)

acc\_random\_forest = round(random\_forest.score(X\_train, y\_train) \* 100, 2)

* **Logistic Regression**:  **Logistic regression** is a statistical **model** that in its basic form uses a **logistic** function to **model** a binary dependent variable and is one of the most popular models used to resolve classification Problems.

*#Logistic Regression*

logreg = LogisticRegression()

logreg.fit(X\_train, y\_train)

Y\_pred = logreg.predict(X\_test)

acc\_log = round(logreg.score(X\_train, y\_train) \* 100, 2)

* **K-Nearest Neighbor:** The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. **It** is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). **KNN** has been **used in** statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique

*#K Nearest Neighbor:*

knn = KNeighborsClassifier(n\_neighbors = 3)

knn.fit(X\_train, y\_train)

Y\_pred = knn.predict(X\_test)

acc\_knn = round(knn.score(X\_train, y\_train) \* 100, 2)

* Gaussian Naïve Bayes: **Gaussian Naive Bayes** is a variant of **Naive Bayes** that follows **Gaussian** normal distribution and supports continuous data.Naive Bayes are a group of supervised machine learning **classification** algorithms based on the **Bayes** theorem. It is a simple **classification** technique, but has high functionality. **Naive Bayes** is a classification algorithm for binary (two-class) and multi-class classification problems. The technique is easiest to understand when described using binary or categorical input values. They find use when the dimensionality of the inputs is high. Complex classification problems can also be implemented by using it.

*#Gaussian Naive Bayes:*

gaussian = GaussianNB()

gaussian.fit(X\_train, y\_train)

Y\_pred = gaussian.predict(X\_test)

acc\_gaussian = round(gaussian.score(X\_train, y\_train) \* 100, 2)

* Decision Tree: Decision tree learning is one of the predictive modelling approaches used in statistics, data mining and machine learning. It uses a decision tree to go from observations about an item to conclusions about the item's target value. builds **classification or regression** models in the form of a **tree** structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated **decision tree** is incrementally developed. Decision trees can handle both categorical and numerical data.The decision tree tool is used in real life in many areas, such as engineering, civil planning, law, and business.

*#Decision Tree*

decision\_tree = DecisionTreeClassifier()

decision\_tree.fit(X\_train, y\_train)

Y\_pred = decision\_tree.predict(X\_test)

acc\_decision\_tree = round(decision\_tree.score(X\_train, y\_train) \* 100, 2)

* Looping to find the best Model

results = pd.DataFrame({

'Model': ['KNN', 'Logistic Regression',

'Random Forest', 'Naive Bayes',

'Decision Tree'],

'Score': [ acc\_knn, acc\_log,

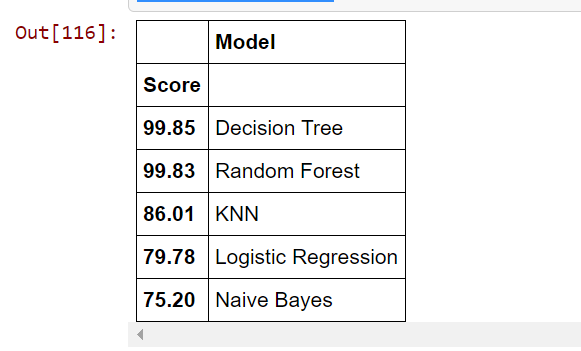
acc\_random\_forest, acc\_gaussian,

acc\_decision\_tree]})

result\_df = results.sort\_values(by='Score', ascending=**False**)

result\_df = result\_df.set\_index('Score')

result\_df.head(9)



As we can see both Random forest and decison tree has performed good, however as per above screenshot the decision tree classifier goes on the first place. But first, let us check, how decision tree,random forest performs, when we use cross validation.

**K-fold cross validation with Random forest**

**from** **sklearn.model\_selection** **import** cross\_val\_score

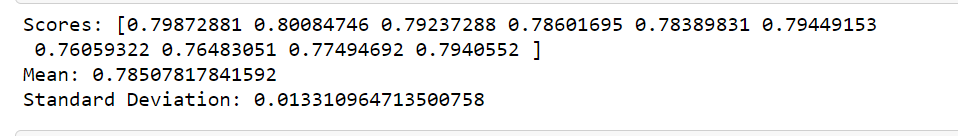
rf = RandomForestClassifier(n\_estimators=100)

scores = cross\_val\_score(rf, X\_train, y\_train, cv=10, scoring = "accuracy")

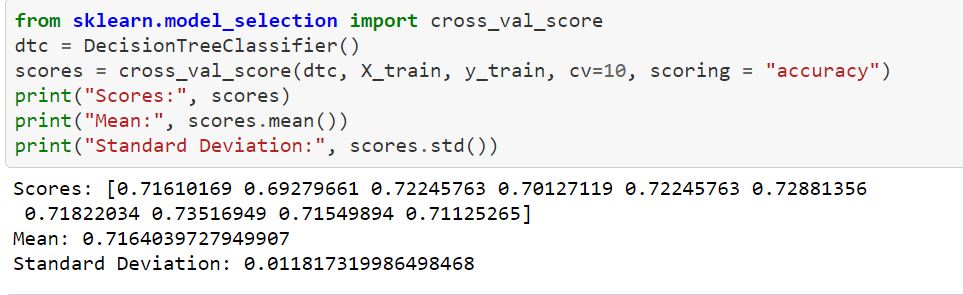
print("Scores:", scores)

print("Mean:", scores.mean())

print("Standard Deviation:", scores.std())



**K-fold cross validation with decison tree**



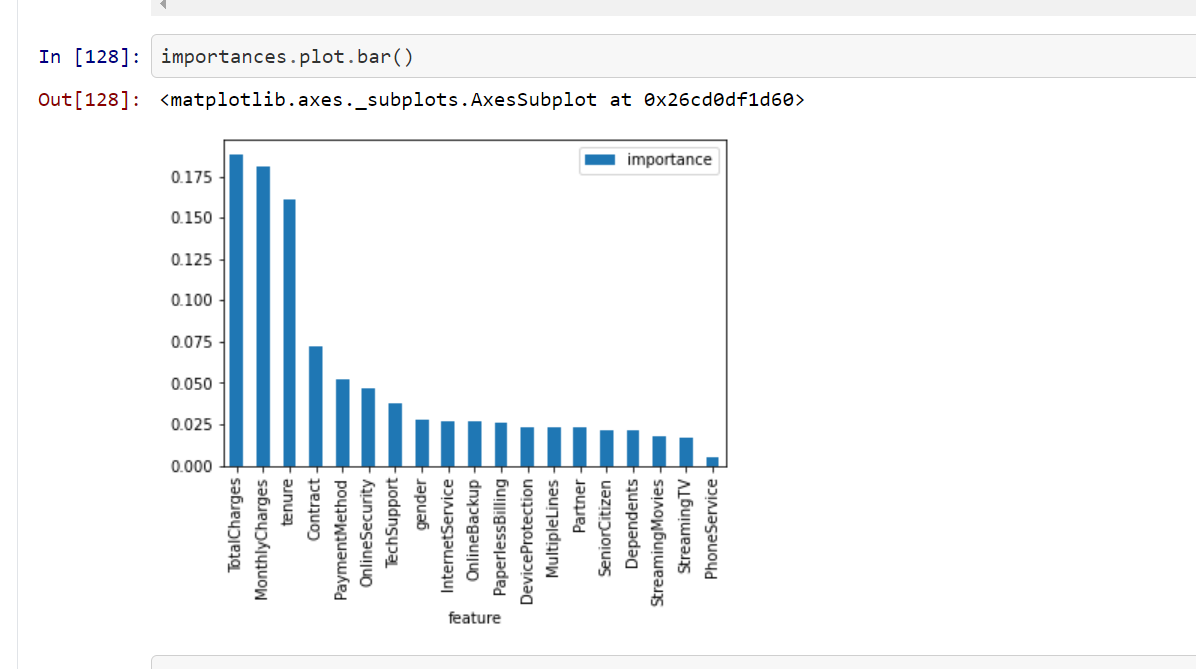
*After cross validation we can see random forest is performing better with score of 78% and standard deviation of .01 %*

**Peek at Feature importance**

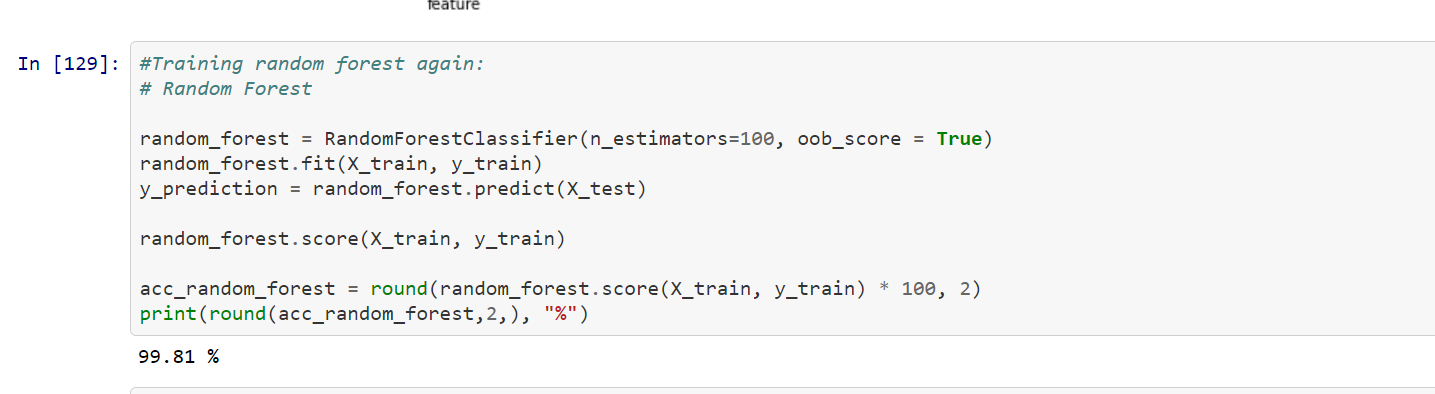
importances = pd.DataFrame({'feature':X\_train.columns,'importance':np.round(random\_forest.feature\_importances\_,3)})

importances = importances.sort\_values('importance',ascending=**False**).set\_index('feature')

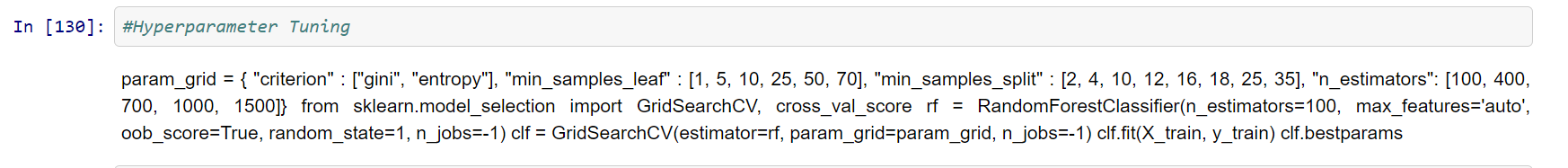
importances.head(15)

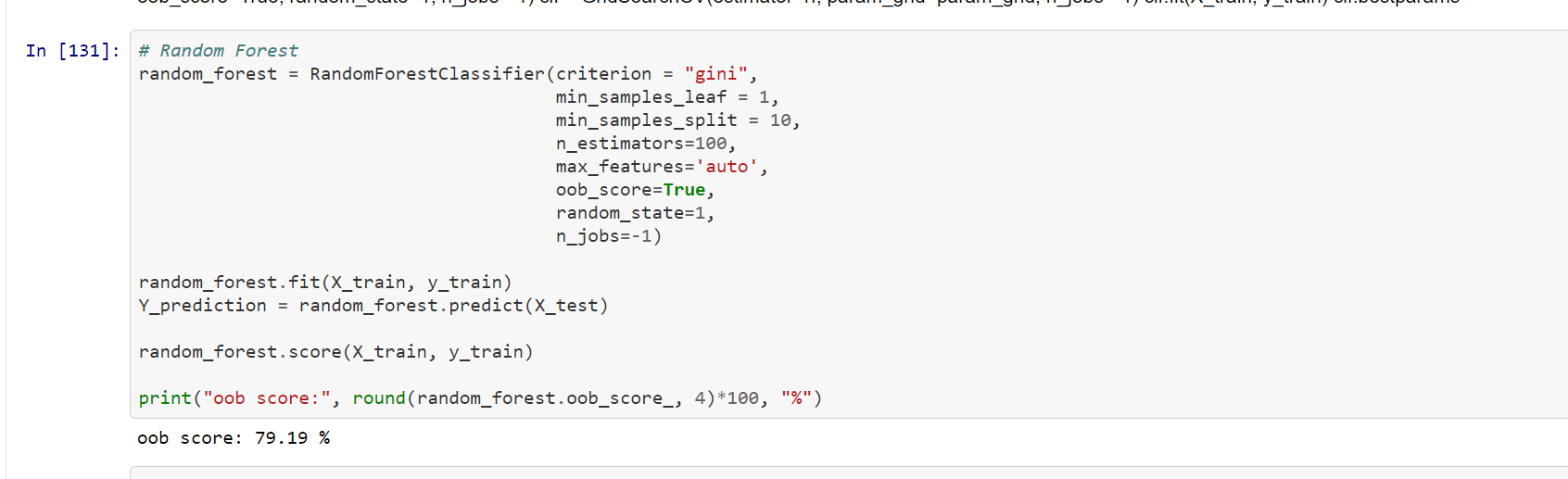


* **Training the Random Forest further**

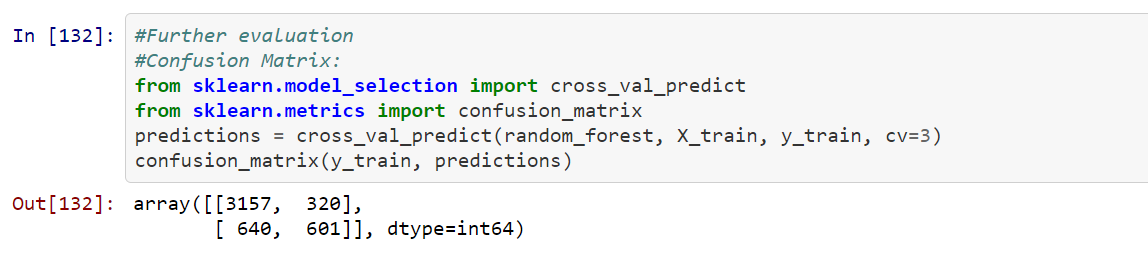


* **Hyperparameter tuning**





* **Further evaluation method through confusion Matrix and F-1 Score**



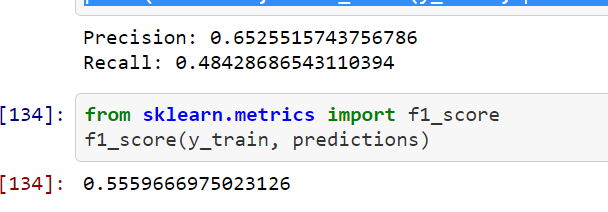
* **Precision and Recall**

*#Precision and Recall:*

**from** **sklearn.metrics** **import** precision\_score, recall\_score

print("Precision:", precision\_score(y\_train, predictions))

print("Recall:",recall\_score(y\_train, predictions))



* **ROC-AUC Curve**

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

false\_positive\_rate, true\_positive\_rate, thresholds = roc\_curve(y\_train, predictions) *# plotting them against each other*

**def** plot\_roc\_curve(false\_positive\_rate, true\_positive\_rate, label=**None**): plt.plot(false\_positive\_rate, true\_positive\_rate, linewidth=2, label=label) plt.plot([0, 1], [0, 1], 'r', linewidth=4)

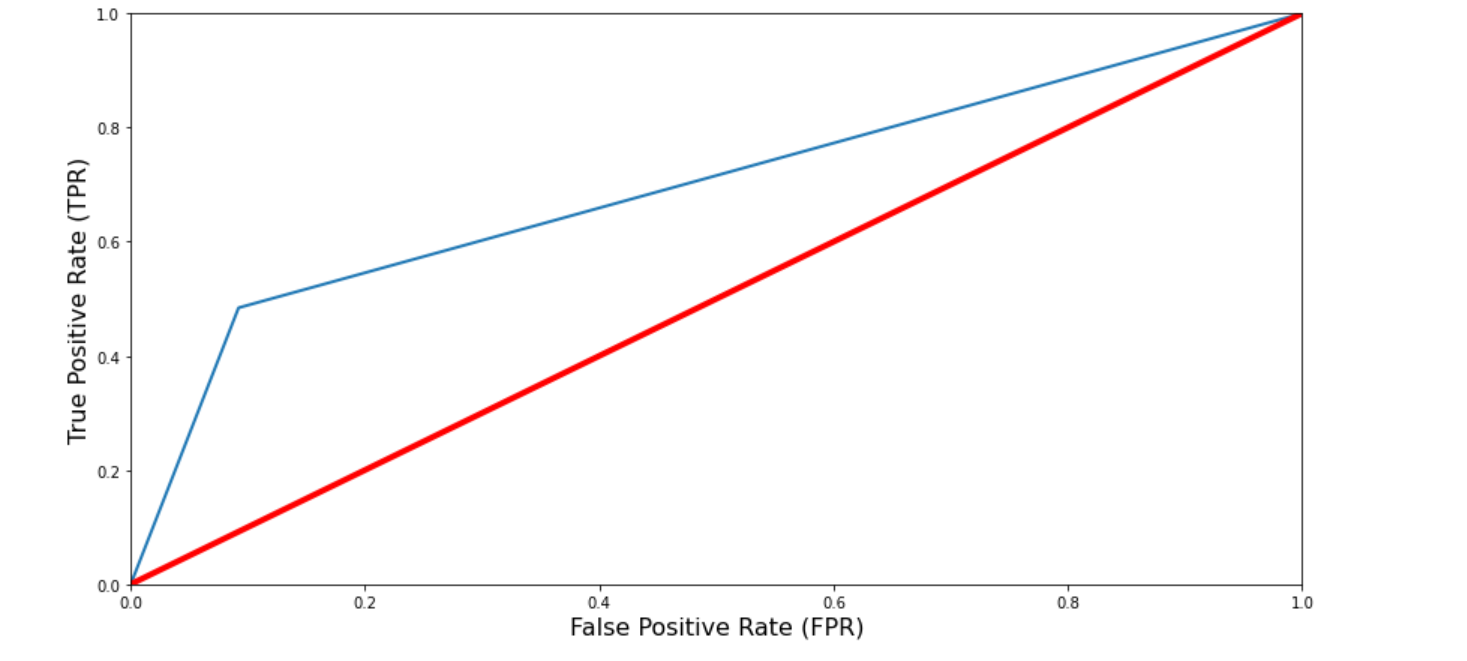
plt.axis([0, 1, 0, 1])

plt.xlabel('False Positive Rate (FPR)', fontsize=16) plt.ylabel('True Positive Rate (TPR)', fontsize=16)

plt.figure(figsize=(14, 7))

plot\_roc\_curve(false\_positive\_rate, true\_positive\_rate)

plt.show()



**from** **sklearn.metrics** **import** roc\_auc\_score

r\_a\_score = roc\_auc\_score(y\_train, predictions)

print("ROC-AUC-Score:", r\_a\_score)



*#Model Saving*

**import** **pickle**

filename='customerchurn.pkl'

pickle.dump(random\_forest,open(filename,'wb'))

**Conclusion:**

Random forest appears to be the best as per the validations performed above.We have also seen in the document that how churn is correlated with various features in the dataset.Totalcharges,monthlycharges,tenure,contract,payment method,online security,backup,techsupport,internet service are among the features which affect the churning the most. The score shows us that in 78% of the cases our model predicted the right outcome for our binary classification problem. That’s considered quite good for a first run, especially when we look which impact each variable has and if that makes sense. So with the final objective to reduce churn and take the right preventing actions in timethe churn can be prevented

a=np.array(y\_test)

predicted=np.array(random\_forest.predict(X\_test))

df\_com=pd.DataFrame({"Original":a,"predicted":predicted},index=range(len(a)))

df\_com

