**Customer Churn Analysis**

**Problem Definition-**

Churn rate, when applied to a customer base, refers to the proportion of contractual customers or subscribers who leave a supplier during a given time period. It is a possible indicator of customer dissatisfaction, cheaper and/or better offers from the competition, more successful sales and/or marketing by the competition, or reasons having to do with the customer life cycle

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.

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.

One of the most famous and useful case studies of churn prediction is in the telecom industry. It is important for telecom companies to analyze all relevant customer data and develop a robust and accurate Churn Prediction model to retain customers and to form strategies for reducing customer attrition rates.

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

You will examine customer data from IBM Sample Data Sets with the aim of building and comparing several customer churn prediction models.

Note: we can find the dataset in the link below.

Downlaod Files:

<https://github.com/dsrscientist/DSData/blob/master/Telecom_customer_churn.csv>

**Attributes Information**

**Prediction column: Target Veriable**

**Churn: Whether the customer churned or not (Yes or No)**

**Two numerical columns:**

1. MonthlyCharges: The amount charged to the customer monthly

2. TotalCharges: The total amount charged to the customer

**Eighteen categorical columns:**

1. CustomerID: Customer ID unique for each customer

2. gender: Whether the customer is a male or a female

3. SeniorCitizen: Whether the customer is a senior citizen or not (1, 0)

4. Partner: Whether the customer has a partner or not (Yes, No)

5. Dependents: Whether the customer has dependents or not (Yes, No)

6. Tenure: Number of months the customer has stayed with the company

7. PhoneService: Whether the customer has a phone service or not (Yes, No)

8. MultipleLines: Whether the customer has multiple lines or not (Yes, No, No phone service)

9. InternetService: Customer’s internet service provider (DSL, Fiber optic, No)

10. OnlineSecurity: Whether the customer has online security or not (Yes, No, No internet service)

11. OnlineBackup: Whether the customer has an online backup or not (Yes, No, No internet service)

12. DeviceProtection: Whether the customer has device protection or not (Yes, No, No internet service)

13. TechSupport: Whether the customer has tech support or not (Yes, No, No internet service)

14. StreamingTV: Whether the customer has streaming TV or not (Yes, No, No internet service)

15. StreamingMovies: Whether the customer has streaming movies or not (Yes, No, No internet service)

16. Contract: The contract term of the customer (Month-to-month, One year, Two years)

17. PaperlessBilling: Whether the customer has paperless billing or not (Yes, No)

18. PaymentMethod: The customer’s payment method (Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic))

**The project is structured as follows:**

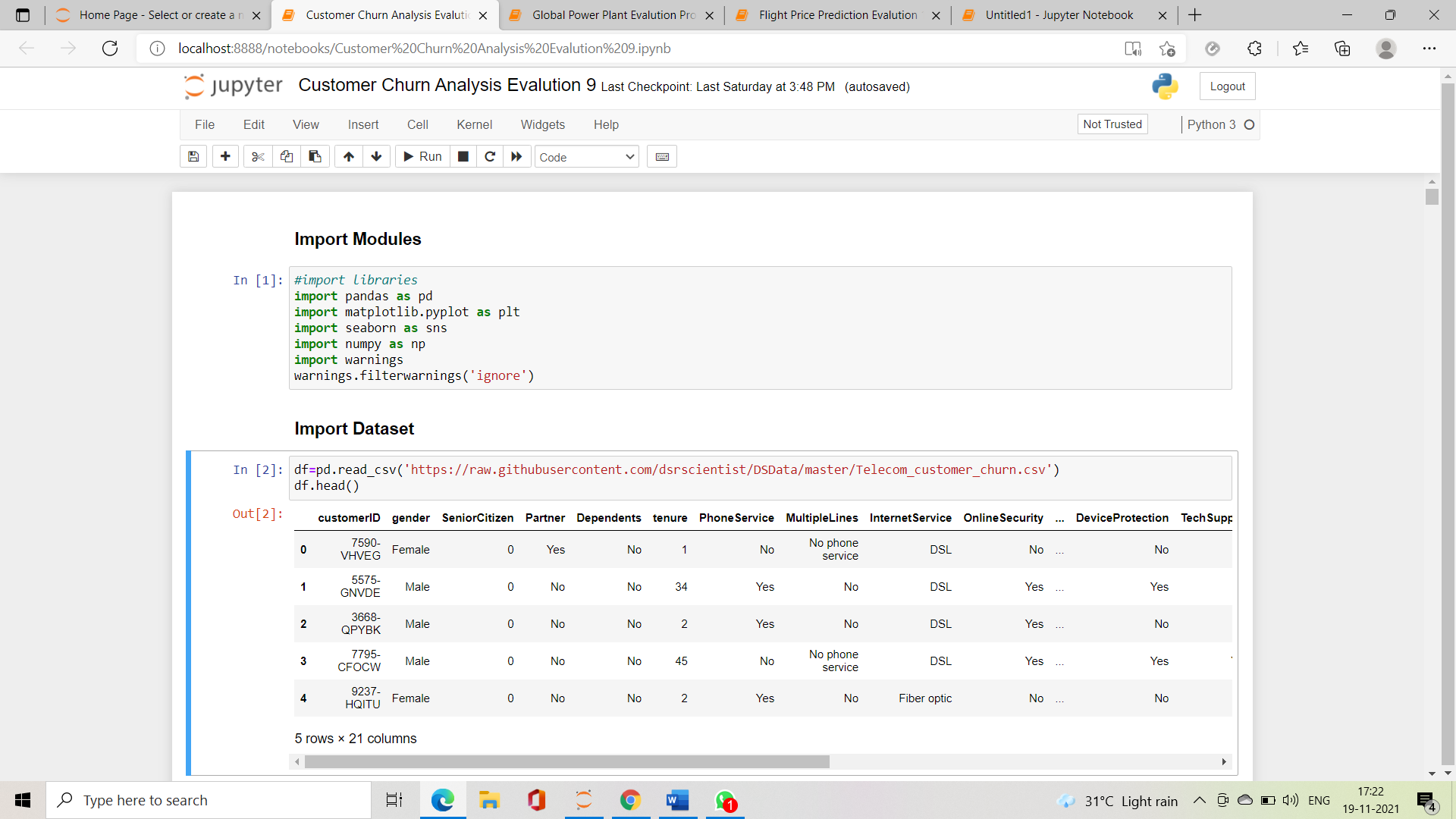
* Data cleaning
* Exploratory Data Analysis
* Data Preprocessing
* Encoding
* Oversampling Technique
* Model Creation and Evaluation
* AUC ROC Curve
* Improving the Model
* Model Saving
* Conclusion

**Data Cleaning**

Start with Importing important libraries:

Head & Tail

To view a small sample of a Series or the DataFrame object, use the head() and the tail() methods. head() returns the first n rows(observe the index values). ... tail() returns the last n rows(observe the index values). The default number of elements to display is five, but you may pass a custom number.

import datset and use head function to show 1st 5row

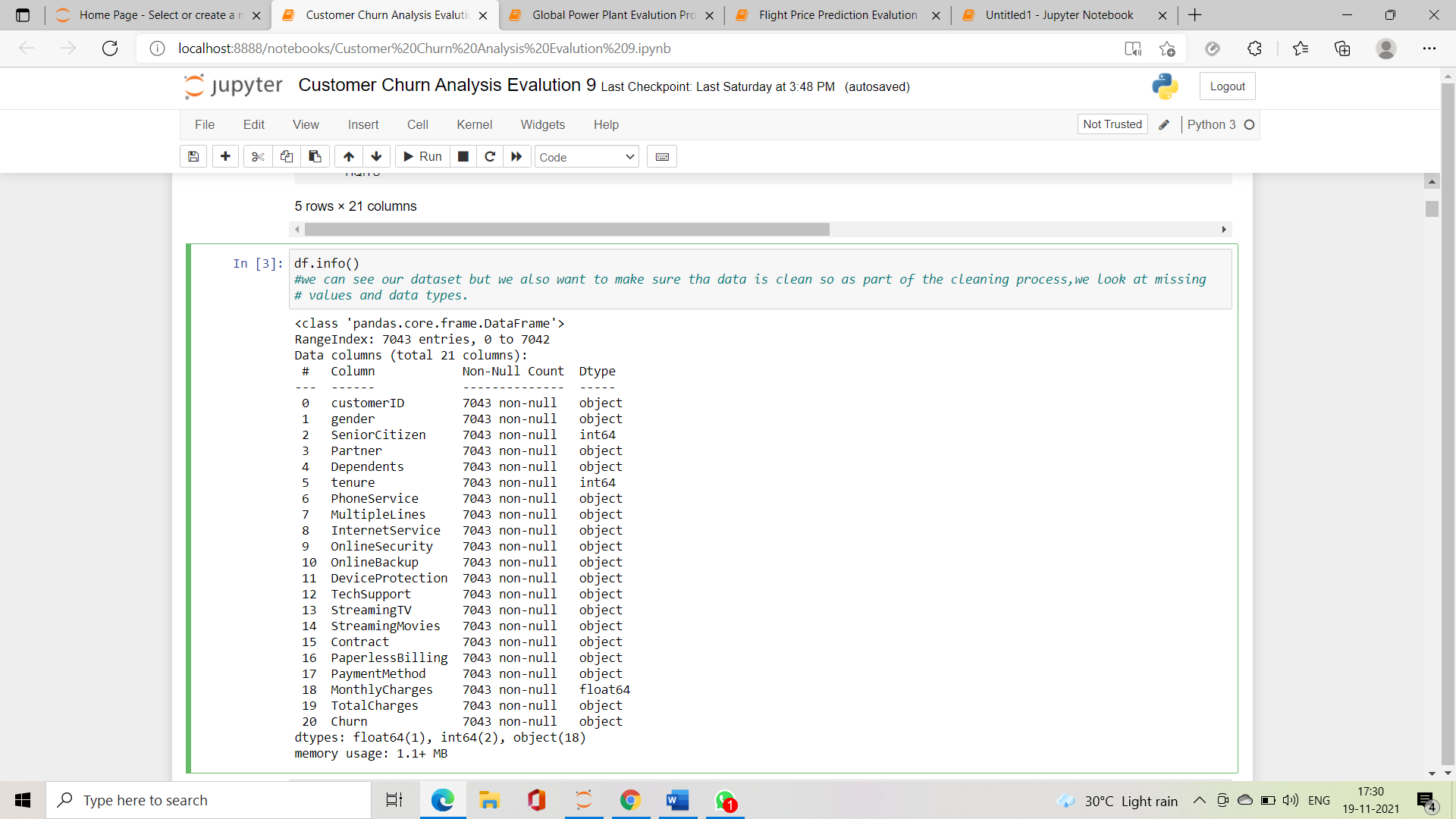
**Exploratory Data Analysis:**

We can see our dataset but we also want to make sure the data is clean, so as part of the cleaning process, we look at missing values and data types.

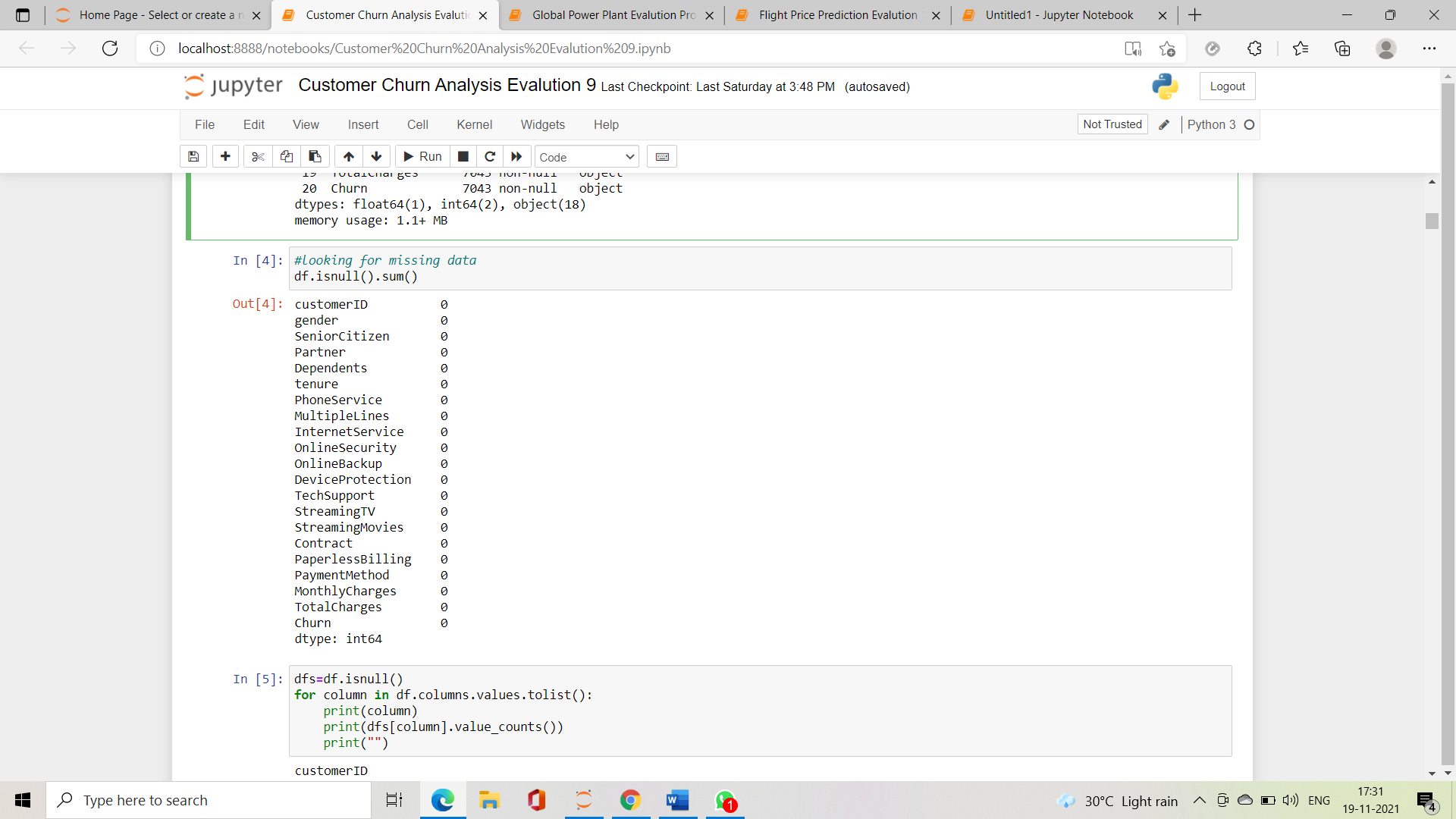
dtype function- A data type object (an instance of numpy. dtype class) describes how the bytes in the fixed-size block of memory corresponding to an array item should be interpreted. It describes the following aspects of the data: Type of the data (integer, float, Python object, etc.)

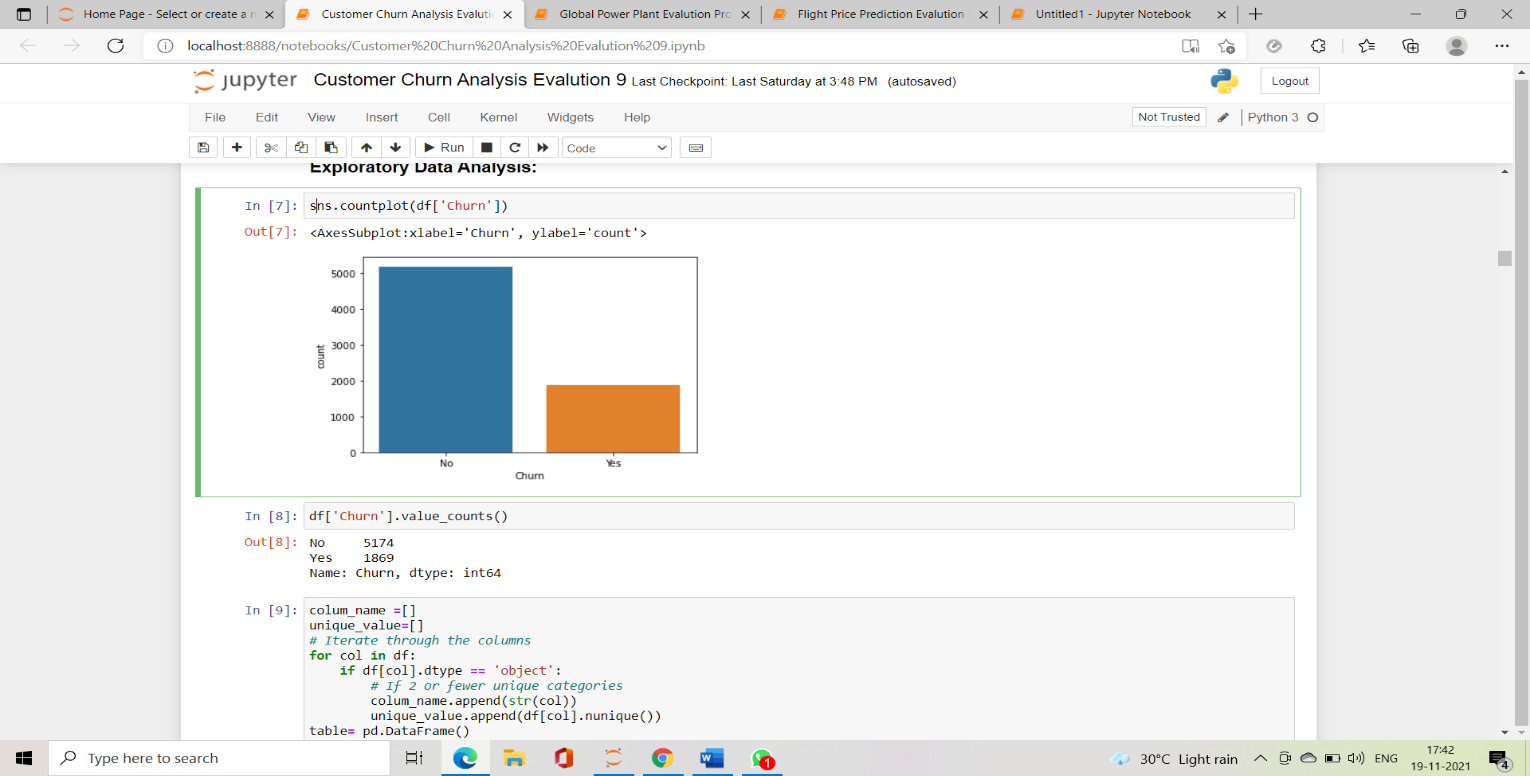
df.info()

we can see our dataset but we also want to make sure tha data is clean so as part of the cleaning process,we look at missing values and data types.



looking for missing data. So our data set don’ have missing values.

df.isnull().sum()

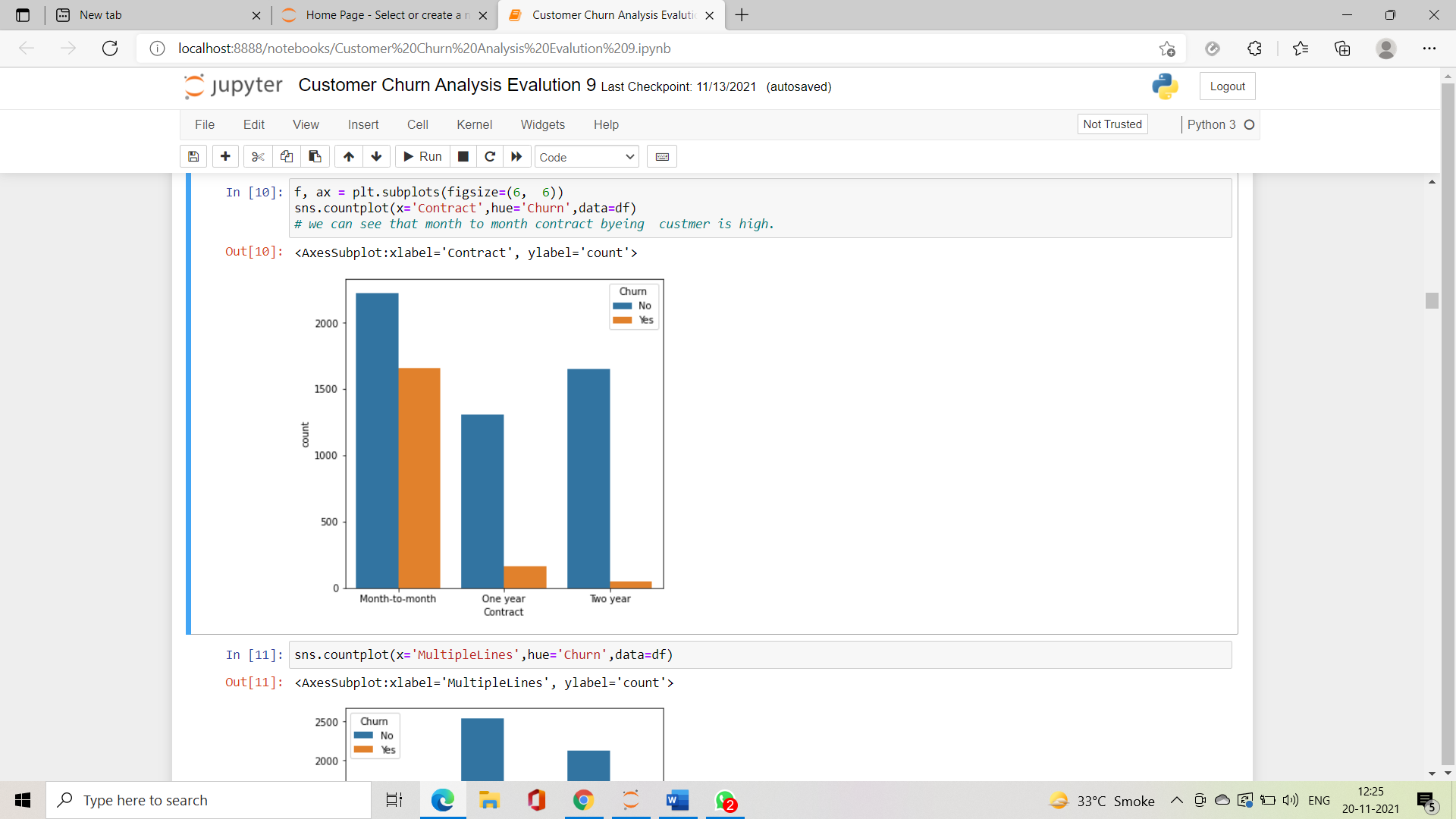
sns.countplot(df['Churn'])

so our dataset is not balanced we need to balanced our data set to get accurate prediction.

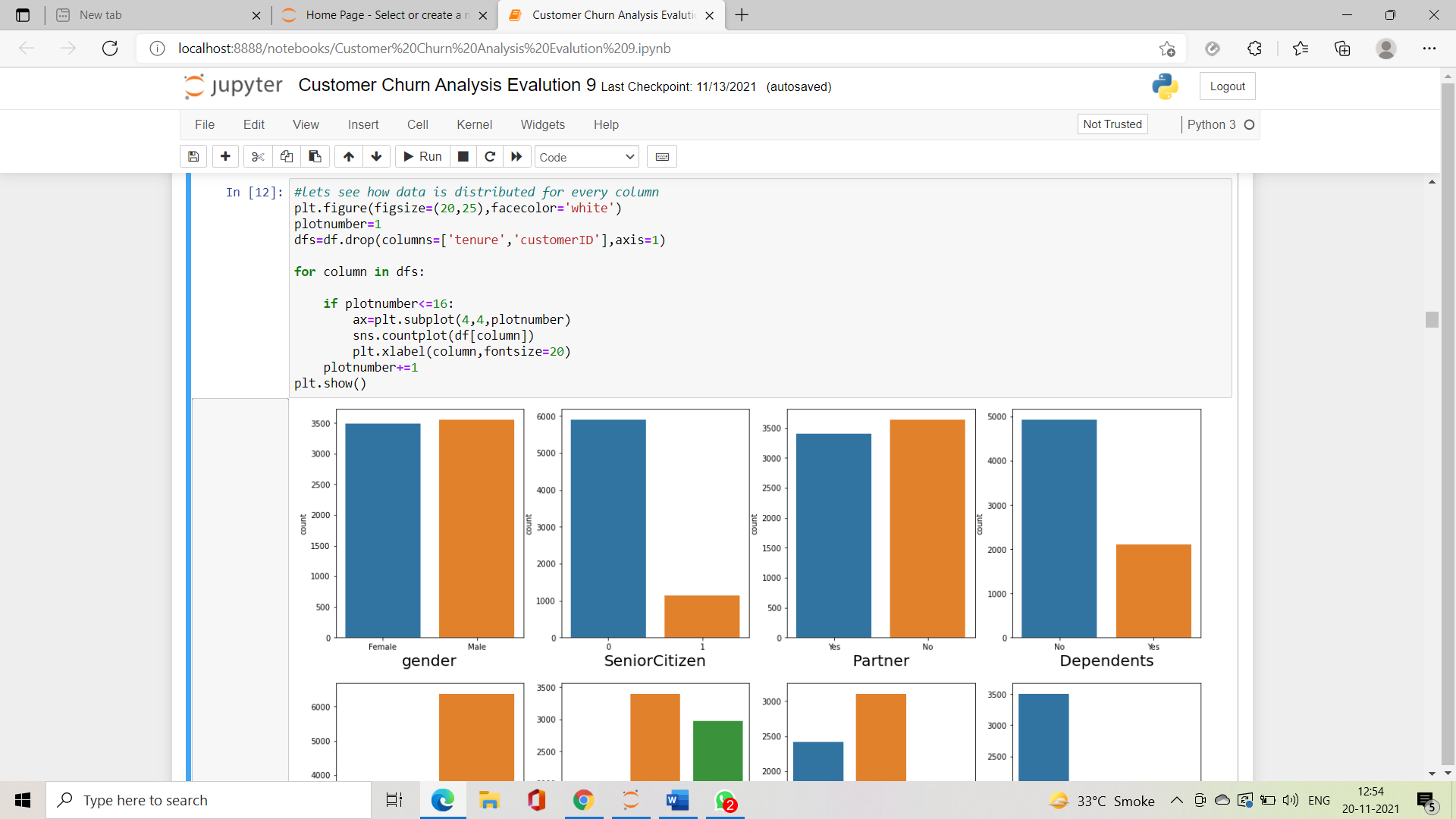
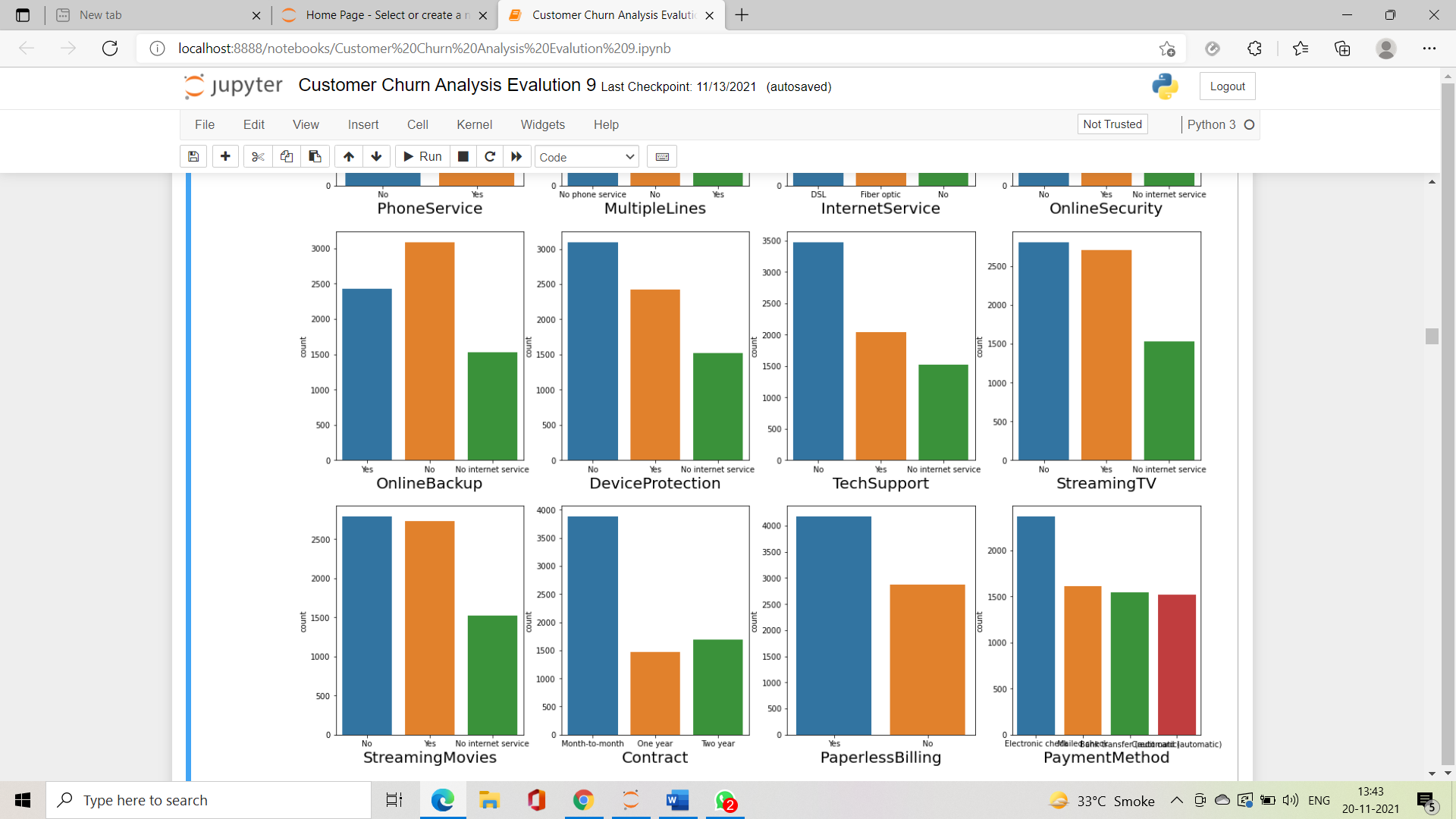
No churn-5174

Yes Churn-1869

**Data Visualization:**

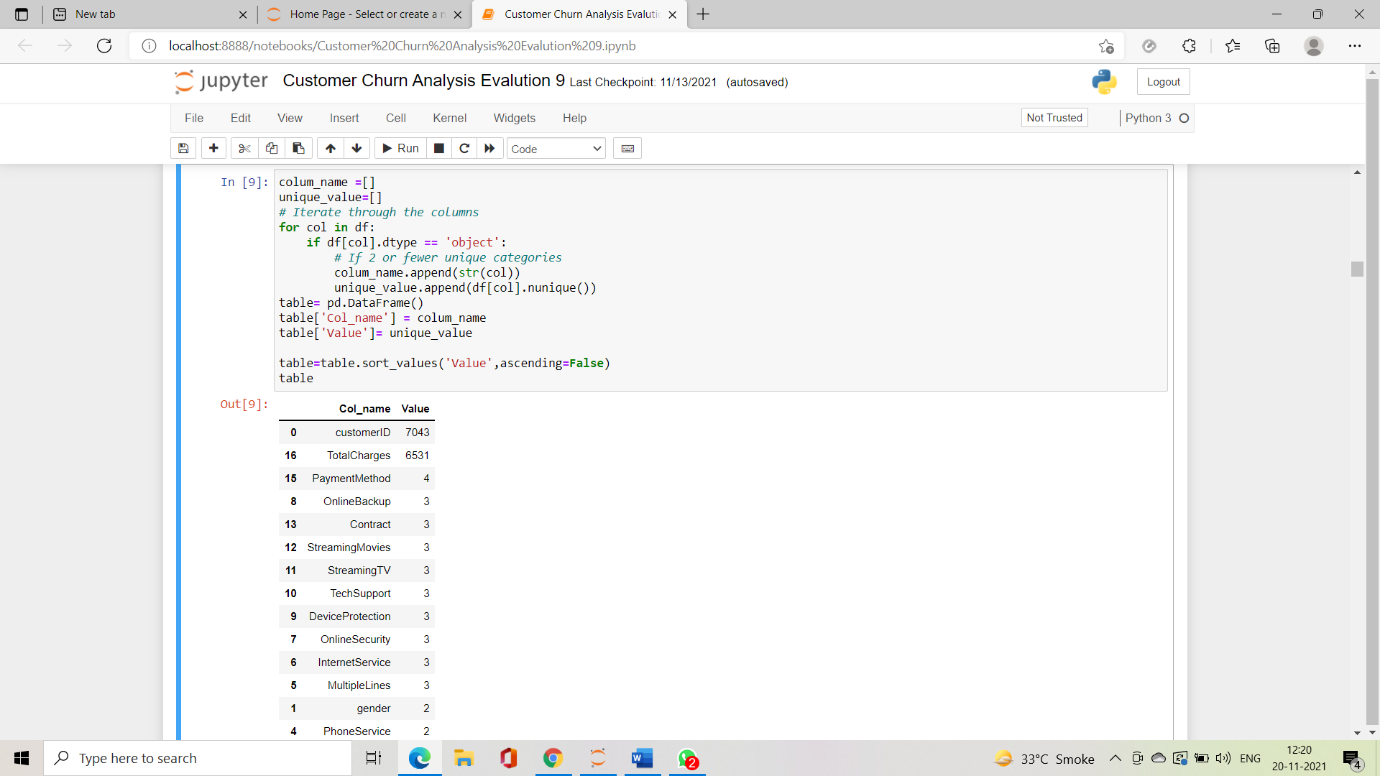
Seaborn is a dataset-oriented library for making statistical representations in Python. Show the counts of observations in each categorical bin using bars with respect to target column.

**Key Observation:**

* Gender Distribution — About half of the customers in our data set are male while the other half are female.almost equal distribution .
* % Senior Citizens — There are only 16% of the customers who are senior citizens. Thus most of our customers in the data are younger people.
* Partner — About 50% of the customers have a partner.
* Dependent status — Only 30% of the total customers have dependents.
* Phone Service — About 90.3% of the customers have phone services.
* Paperless Billing— About 59.2% of the customers make paperless billing
* Few customers don’t have phone service
* Customers with multiple lines have a slightly higher churn rate
* Customers without internet have a very low churn rate
* Customers with fiber are more probable to churn than those with a DSL connection
* Electronic Check is the Largest Payment method
* Electronic Check has most churn in Payment MetLabel Encoder:

A label is actually a number or a string that represents a particular set of entities. Labels helps the model in better understanding of the dataset and enables the model to learn more complex structures. Label Encoder performs the conversion of these labels of categorical data into a numeric format

Before that we need to convert data from categorical to numerical format by using label incoder

* Categorical features
* This dataset has 16 categorical features:
* Six binary features (Yes/No)
* Nine features with three unique values each (categories)
* One feature with four unique values

**objList = df.select\_dtypes(include = "object").columns**

**print (objList)**

Index(['customerID', 'gender', 'Partner', 'Dependents', 'PhoneService',

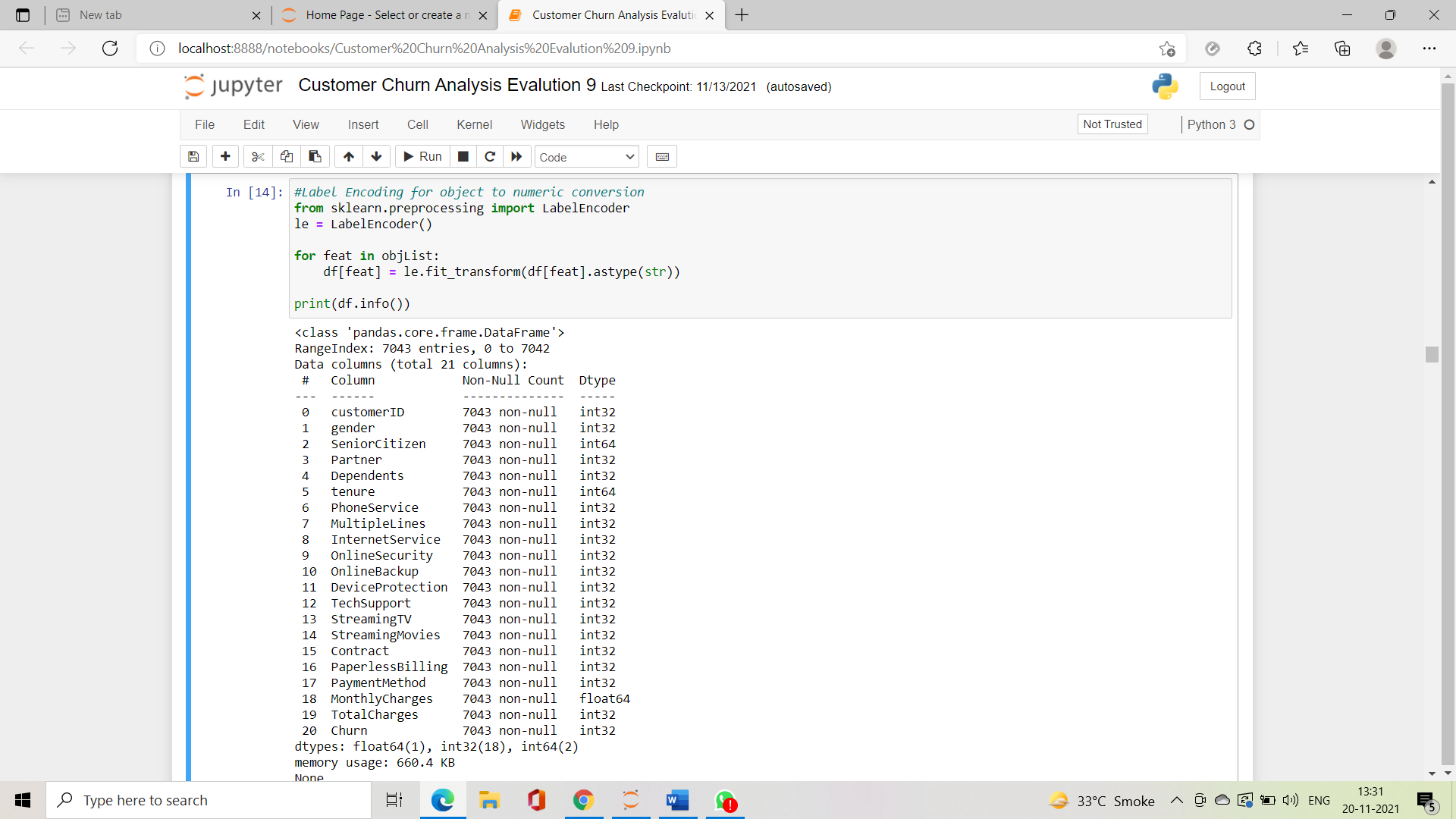
'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup',

'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',

'Contract', 'PaperlessBilling', 'PaymentMethod', 'TotalCharges',

'Churn'],

dtype='object')

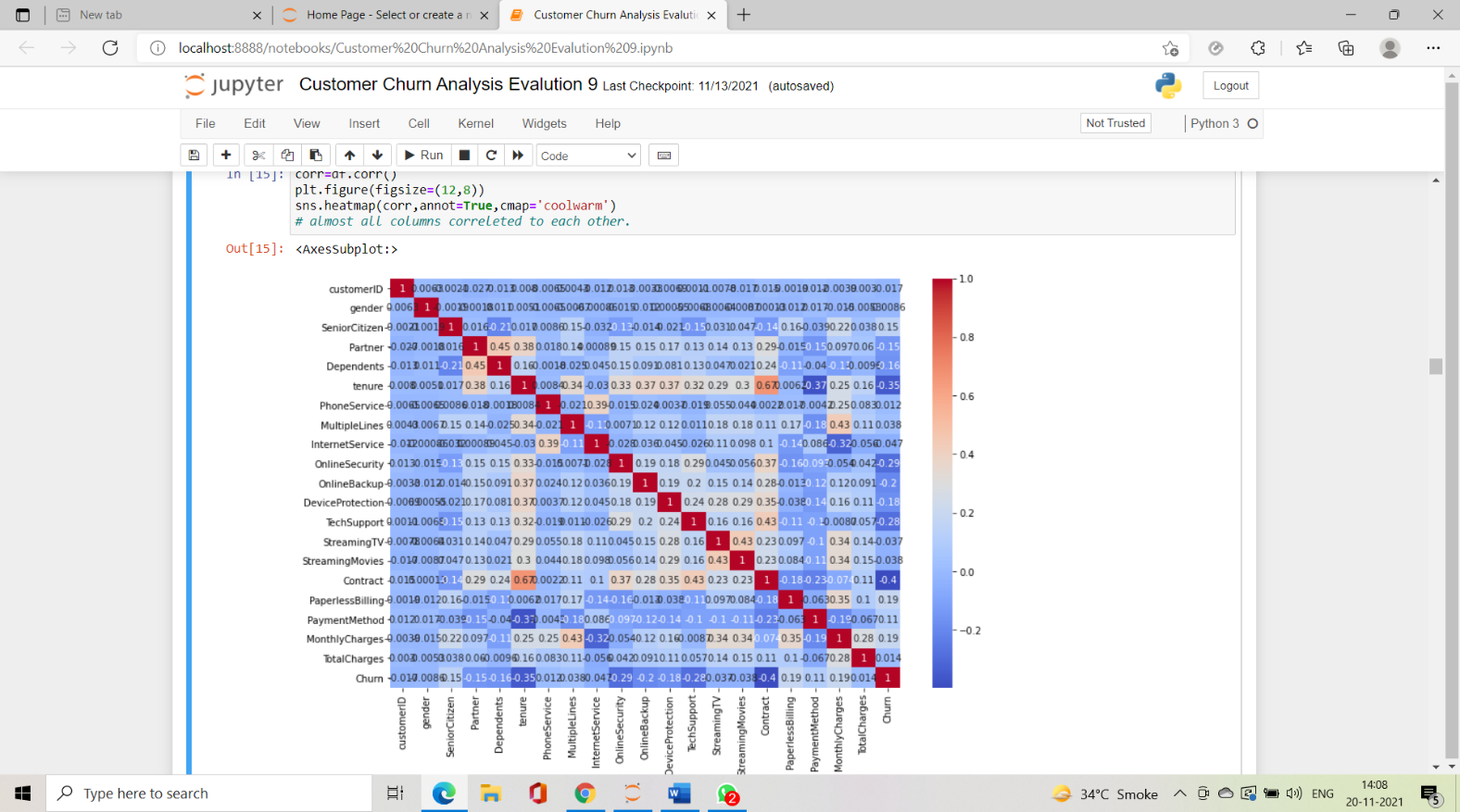
so above column convert to numric format.by using label Encoder.

so now we can see every single column is in int32 format.

**Correlation:**

Correlation is a statistical term describing the degree to which two variables move in coordination with one another. If the two variables move in the same direction, then those variables are said to have a positive correlation. If they move in opposite directions, then they have a negative correlation.

Correlation heatmap is graphical representation of correlation matrix representing correlation between different variables. The value of correlation can take any values from -1 to 1. ... Correlation between two variables can also be determined using scatter plot between these two variables

**Correlation Heatmap:**

**Skewd Data :**

in this dataset don’t have that much of sked data.expect phone services

**To check Distribution of skewness**

Right skewness can be reduced applying following transformation

Square root

Cube root

For right-skewed data—tail is on the right, positive skew—, common transformations include square root, cube root, and log. For left-skewed data—tail is on the left, negative skew—, common transformations include square root (constant – x), cube root (constant – x), and log (constant – x).

df.skew()

customerID 0.000000

gender -0.019031

SeniorCitizen 1.833633

Partner 0.067922

Dependents 0.875199

tenure 0.239540

PhoneService -2.727153

MultipleLines 0.118719

InternetService 0.205423

OnlineSecurity 0.416985

OnlineBackup 0.182930

DeviceProtection 0.186847

TechSupport 0.402365

StreamingTV 0.028486

StreamingMovies 0.014657

Contract 0.630959

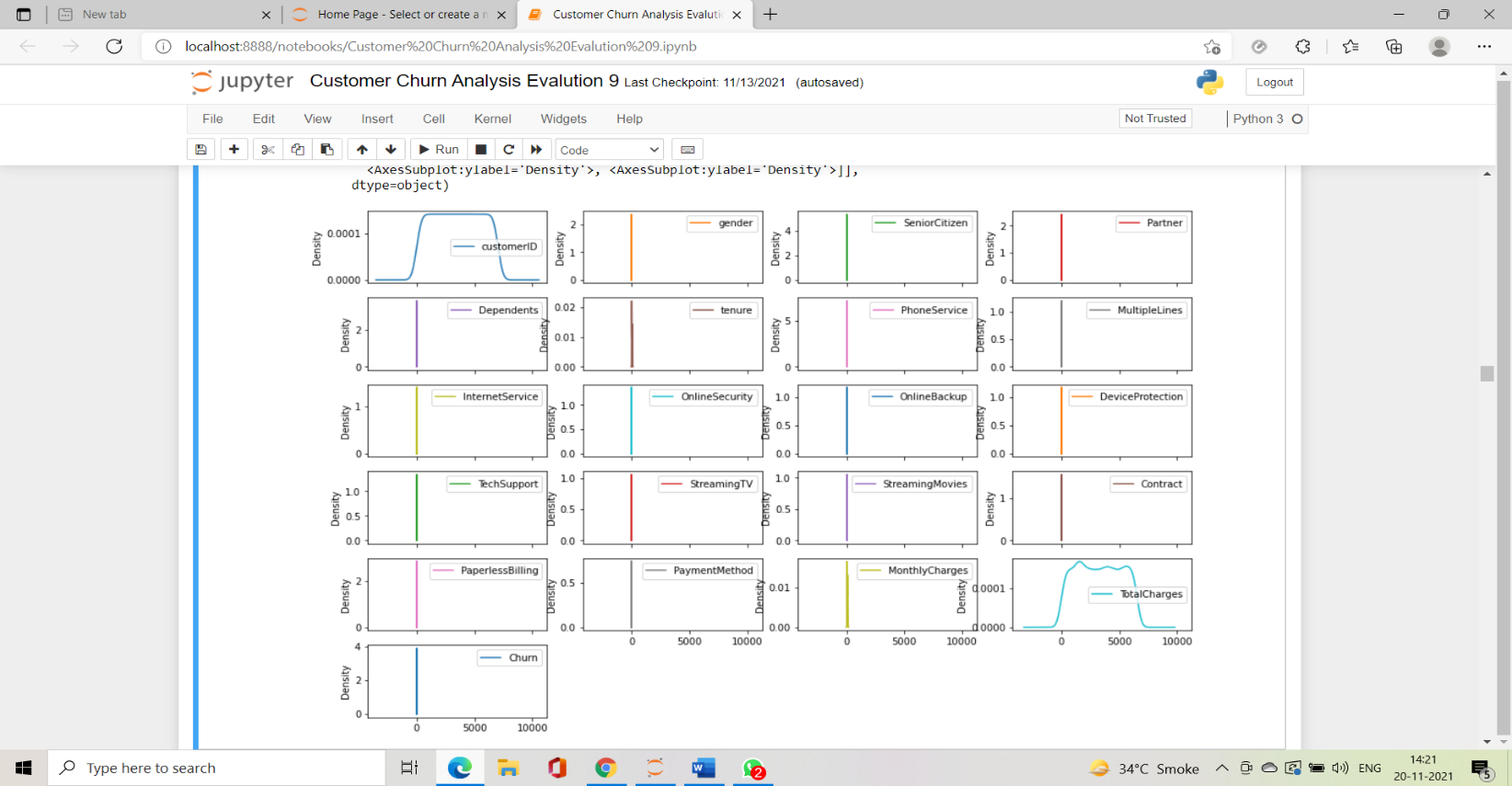
PaperlessBilling -0.375396

PaymentMethod -0.170129

MonthlyCharges -0.220524

TotalCharges 0.015857

Churn 1.063031

dtype: float64**df.plot(kind='kde',subplots=True,layout=(7,4),figsize=(15,12))**

**Outlier Detction:**

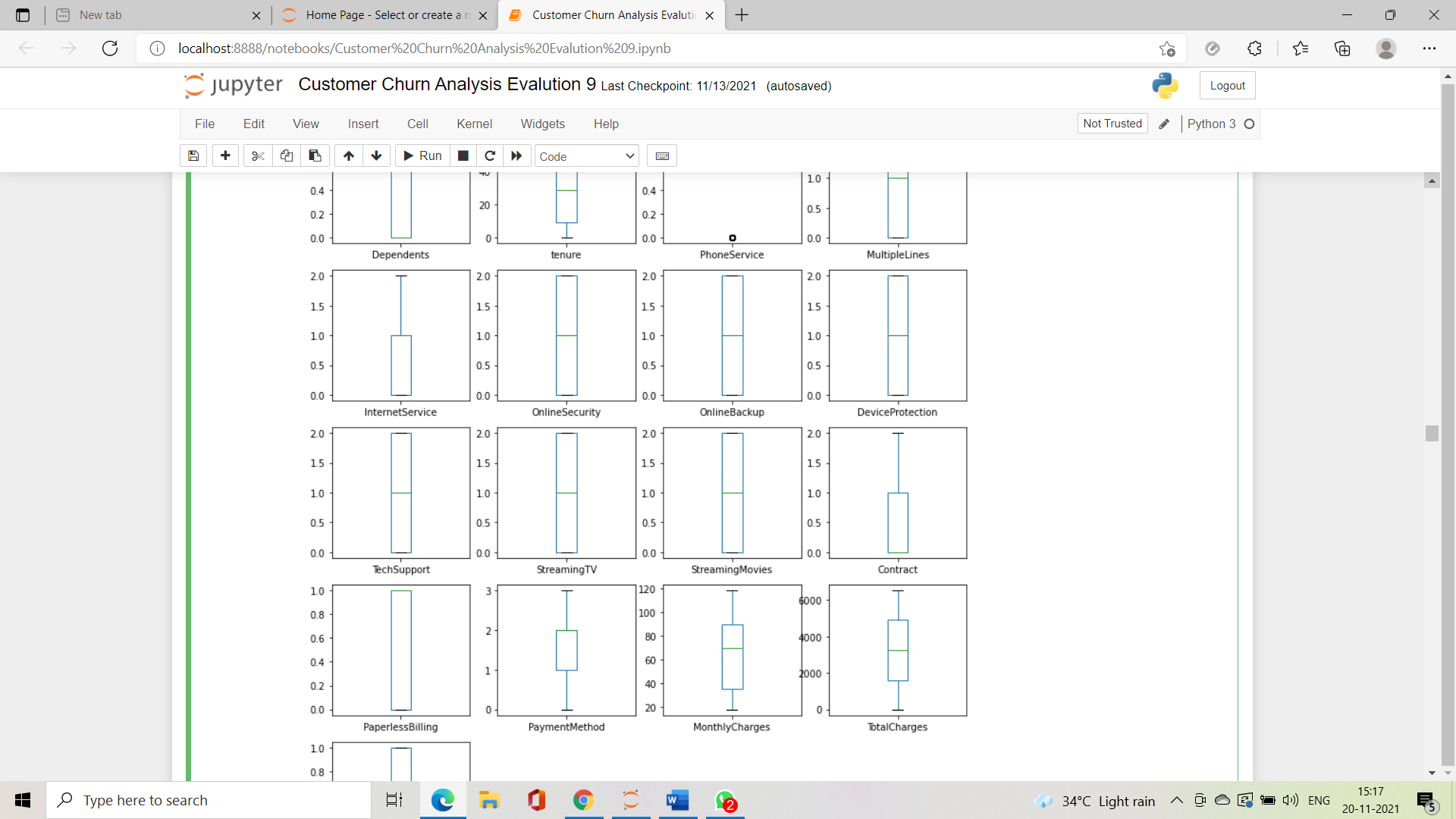
**An outlier is an observation of a data point that lies an abnormal distance**

Z-Score method: In which the distribution of data in the form mean is 0 and the standard deviation (SD) is 1 as Normal Distribution format.

mathematically to find the Outliers as follows Z-Score and Inter Quartile Range (IQR) Score methods

(IQR) Score method: In which data has been divided into quartiles (Q1, Q2, and Q3). Please refer to the picture Outliers Scaling above. Ranges as below.

df.plot(kind='box',subplots=True,layout=(10,4),figsize=(12,30))

#checking Outlier ,so our data is not haveing any outlier .expect phone services,SeniorCitizen

So our data set not contain that much of outliear so we move forward.

Train Test Split:Dividing the dataset into independent and dependent data before scaling

x=df.drop('Churn',axis=1)

y=df['Churn'

**# import libraries for model devloping .**

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

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

**from sklearn.metrics import confusion\_matrix,classification\_report**

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

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

**from sklearn.tree import DecisionTreeClassifier**

**from sklearn.neighbors import KNeighborsClassifier**

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

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

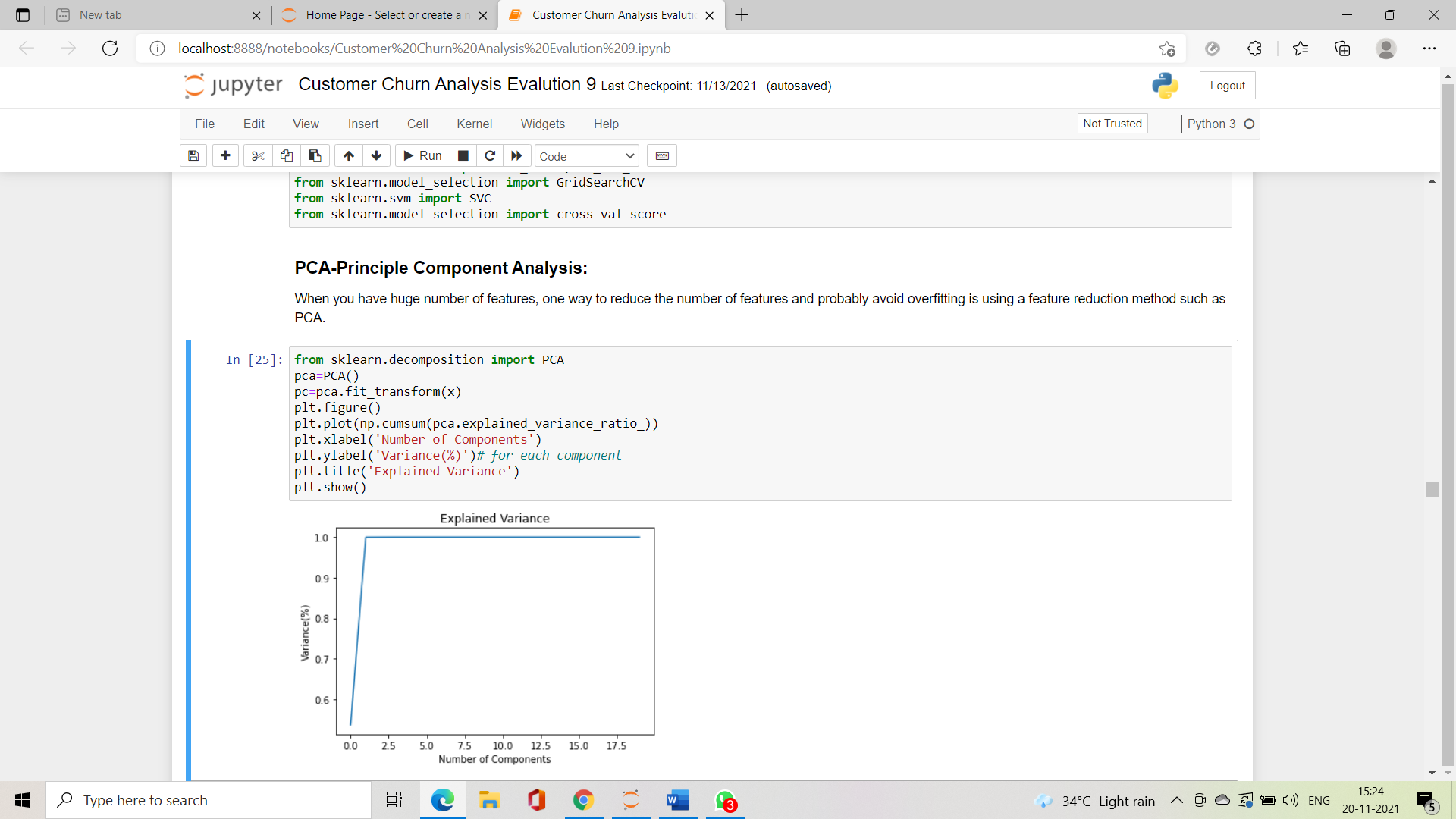
**from sklearn.model\_selection import GridSearchCV**

**from sklearn.svm import SVC**

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

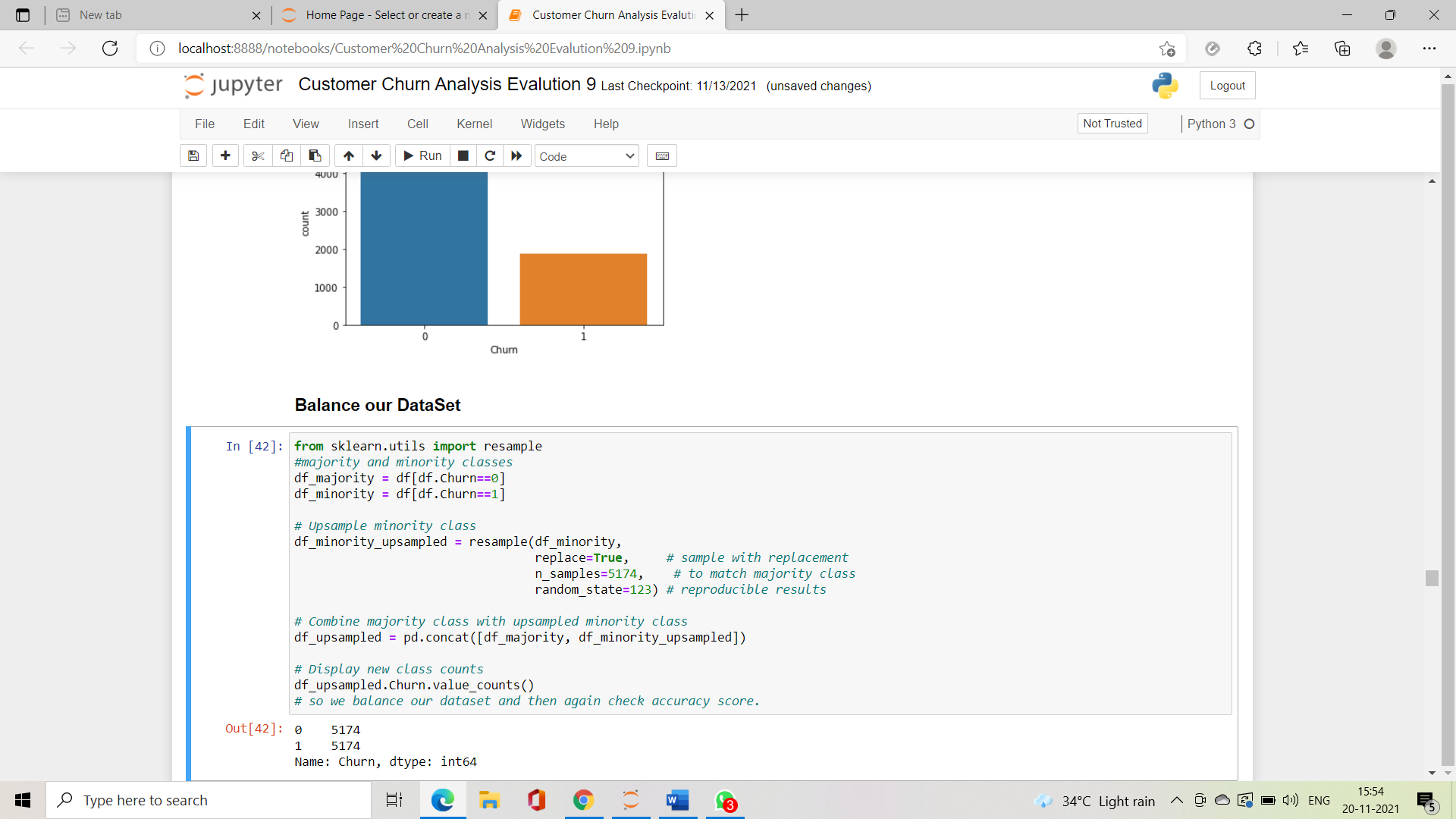
**PCA-Principle Component Analysis:**

**When you have huge number of features, one way to reduce the number of features and probably avoid overfitting is using a feature reduction method such as PCA.**



**Balance our DataSet-**

so we balance our dataset and then again check accuracy score.

Random Oversampling Imbalanced Datasets-Random oversampling involves randomly duplicating examples from the minority class and adding them to the training dataset.

**Standard Scaler**

The next step is to bring the data to a common scale, since there are certain columns with very small values and some columns with high values. This process is important as values on a similar scale allow the model to learn better. We use standard scaler for this process

**from sklearn.preprocessing import StandardScaler**

**scale=StandardScaler()**

**x=scale.fit\_transform(x)**

**x=pd.DataFrame(x)**

**Train Test Split**

**Divides data into Train and Test Subset**

**x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=.20,random\_state=42)**

**print(x.shape,x\_train.shape,x\_test.shape)**

**print(y.shape,y\_train.shape,y\_test.shape)**

**(7043, 20) (5634, 20) (1409, 20)**

**(7043,) (5634,) (1409,)**

**Model**

**For Starter, the LogisticRegression(Classifier) model is implemented to show to results of the basic model and its predictions**

**lg=LogisticRegression()**

**lg.fit(x\_train,y\_train)**

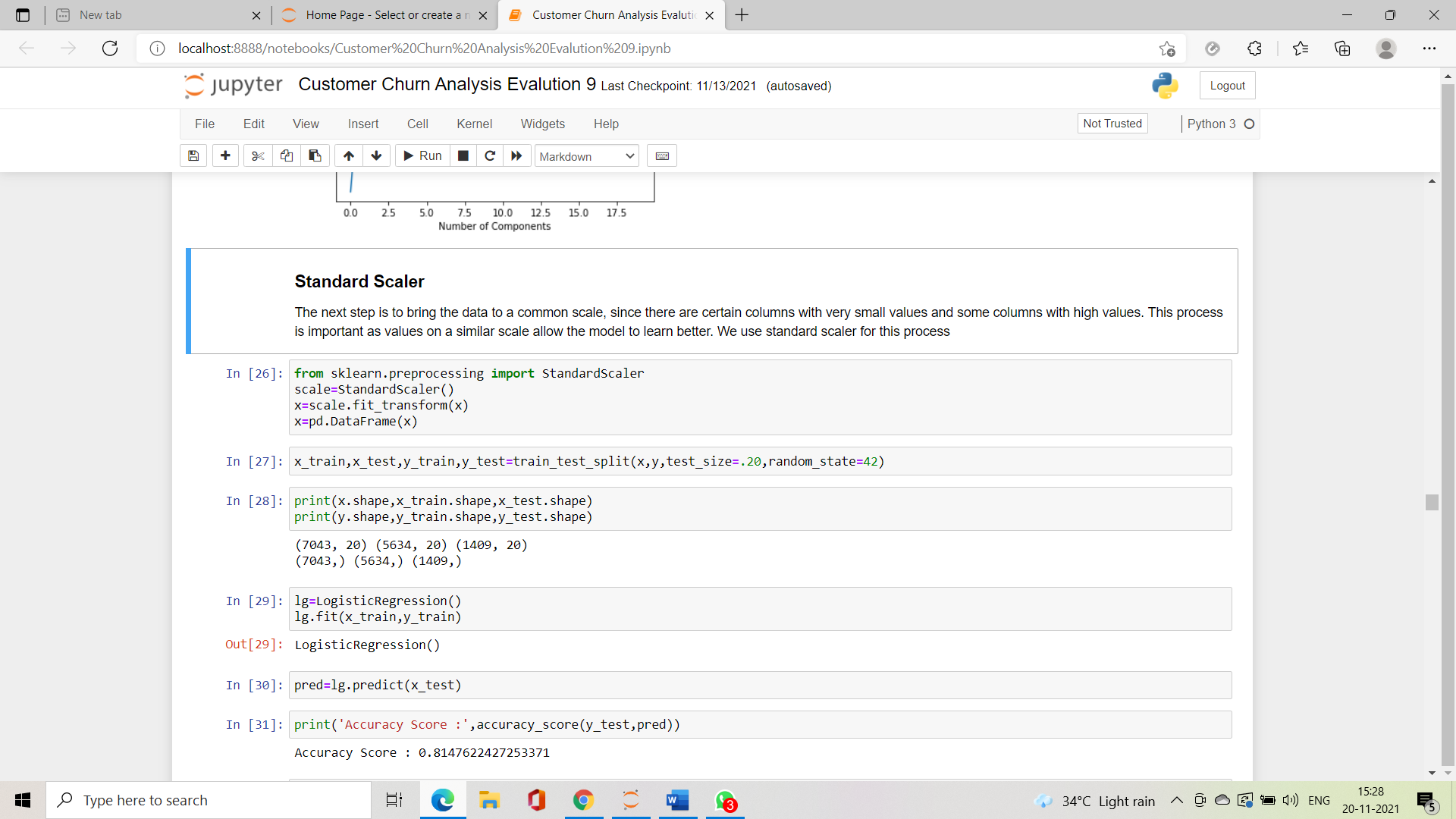
**pred=lg.predict(x\_test)**

**Test Predict**

**Model prediction on the training dataset using logistic Regression()**

**print('Accuracy Score :',accuracy\_score(y\_test,pred))**

**Accuracy Score : 0.8147622427253371**

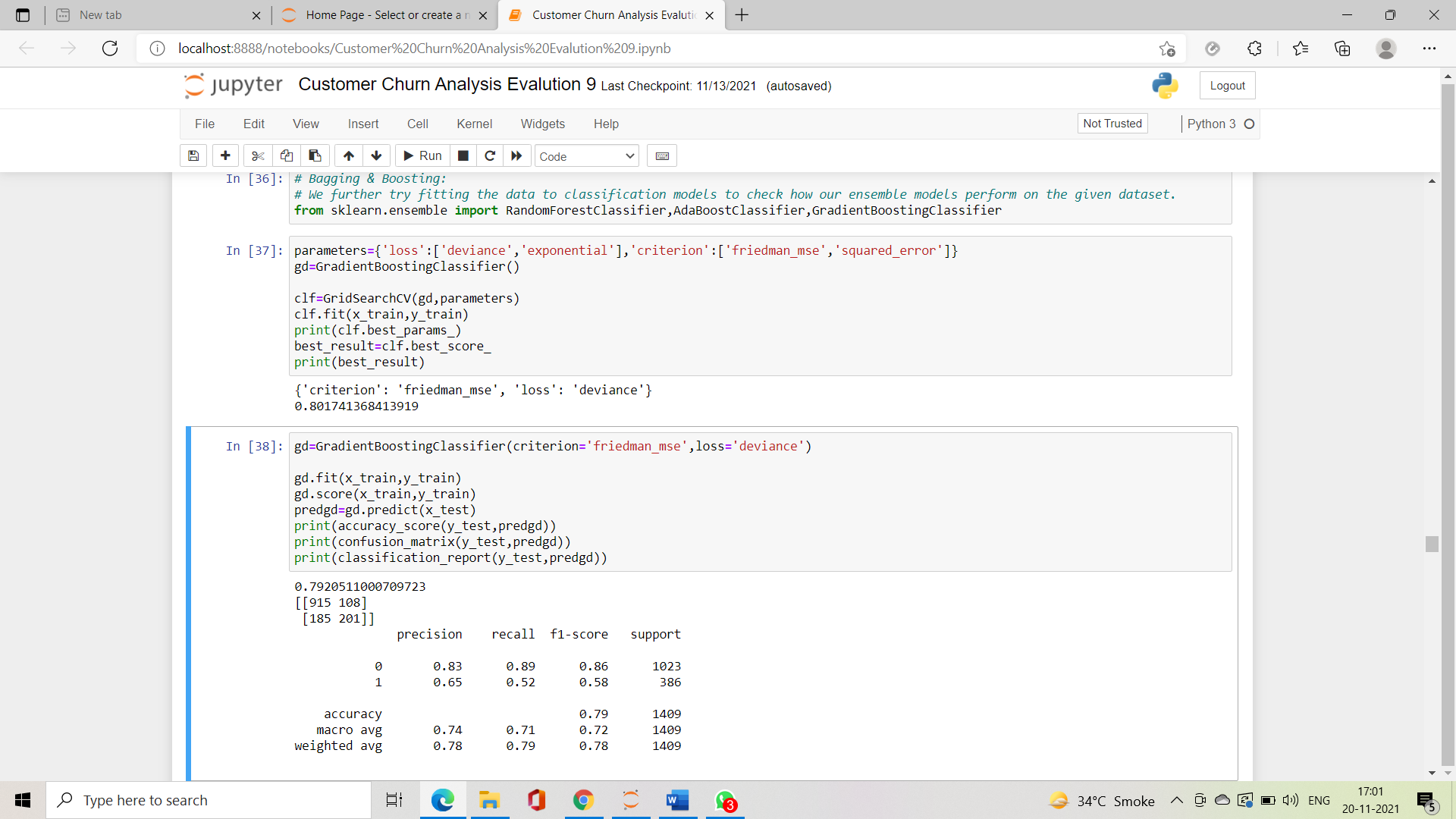
****

**Ensemble methods-**

Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model . To better understand this definition lets take a step back into ultimate goal of machine learning and model building.

**Bagging & Boosting:**

We further try fitting the data to classification models to check how our ensemble models perform on the given dataset.

from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier

**GradientBoostingClassifier() -Model 1st Result**

**Hyperparameter Tuning:GRidSearch CV-80,Model Development-79,Cross validation-80**

**By using Random Forest Classifier Hyperparameter Tunning**

**parameters={'n\_estimators':[100],'criterion':['gini','entropy']}**

**rf=RandomForestClassifier()**

**clf=GridSearchCV(rf,parameters)**

**clf.fit(x\_train,y\_train)**

**print(clf.best\_params\_)**

**best\_result=clf.best\_score\_**

**print(best\_result)**

**{'criterion': 'entropy', 'n\_estimators': 100}**

**0.8834255002408161**

**Model Development-**

**rf=RandomForestClassifier(n\_estimators=100,criterion='gini')**

**#rf.RandomForestClassifier()**

**rf.fit(x\_train,y\_train)**

**rf.score(x\_train,y\_train)**

**predrf=rf.predict(x\_test)**

**print(accuracy\_score(y\_test,predrf))**

**print(confusion\_matrix(y\_test,predrf))**

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

0.9072463768115943

[[899 145]

[ 47 979]]

precision recall f1-score support

0 0.95 0.86 0.90 1044

1 0.87 0.95 0.91 1026

accuracy 0.91 2070

macro avg 0.91 0.91 0.91 2070

weighted avg 0.91 0.91 0.91 2070

We see that the Random Forest Classifier gives us an accuracy of ~88% (higher than gb=GradientBoostingClassifier),and the f1-score, recall and precision scores also improve.Hence we choose ‘Random Forest classifier’ as our final model,and proceed with hypertuning the model.But before this, we perform k-folds cross validation on our dataset

**K-Fold Cross validation-**

Cross-Validation is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model and the other used to validate the model.

#k-fold cross\_validation\_ score

score=cross\_val\_score(rf,x,y,cv=5)

print(score)

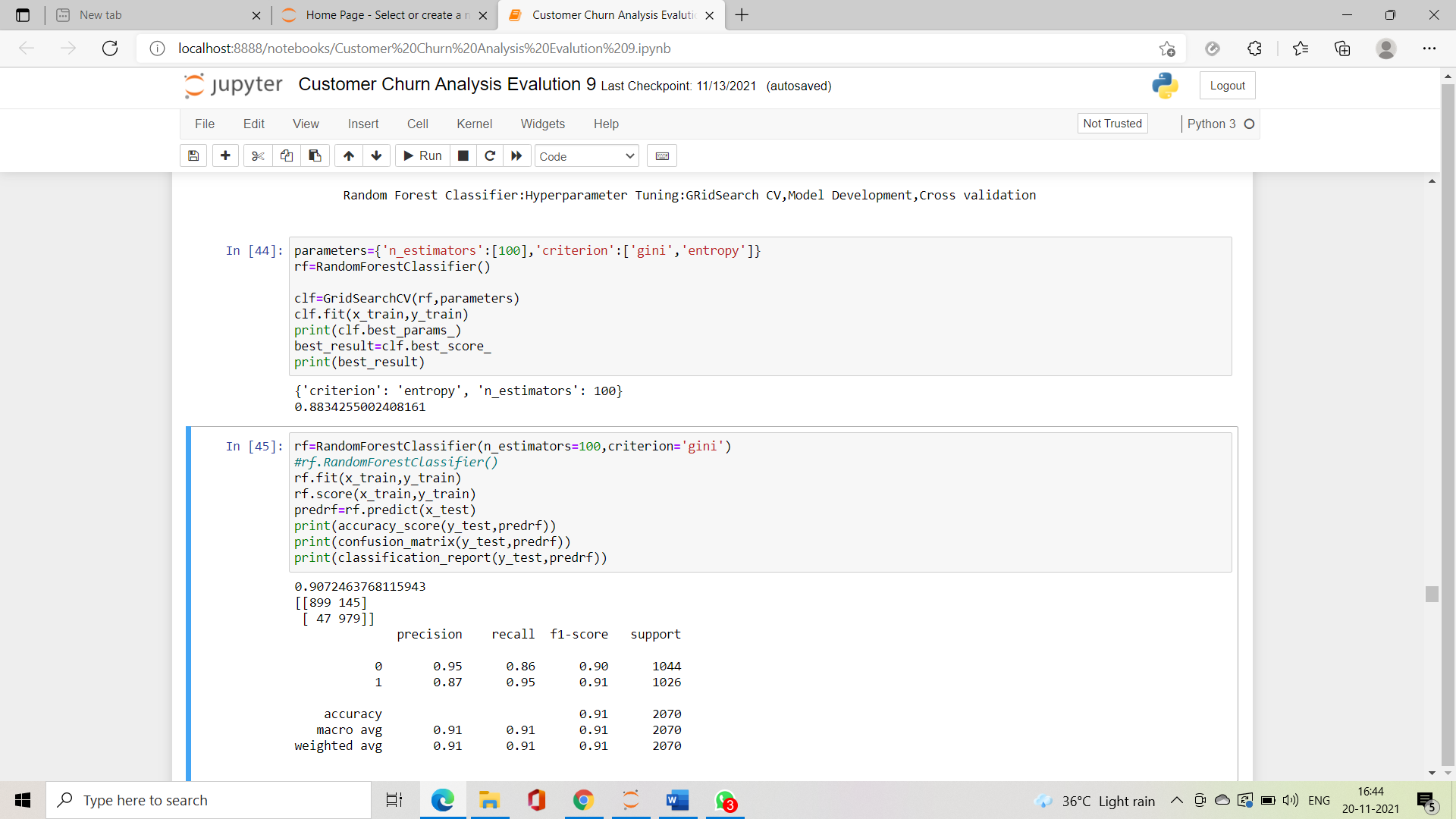
print(score.mean())

print(score.std())

[0.91062802 0.90724638 0.89130435 0.90671822 0.91251812]

0.9056830180044504

0.007502447792684386

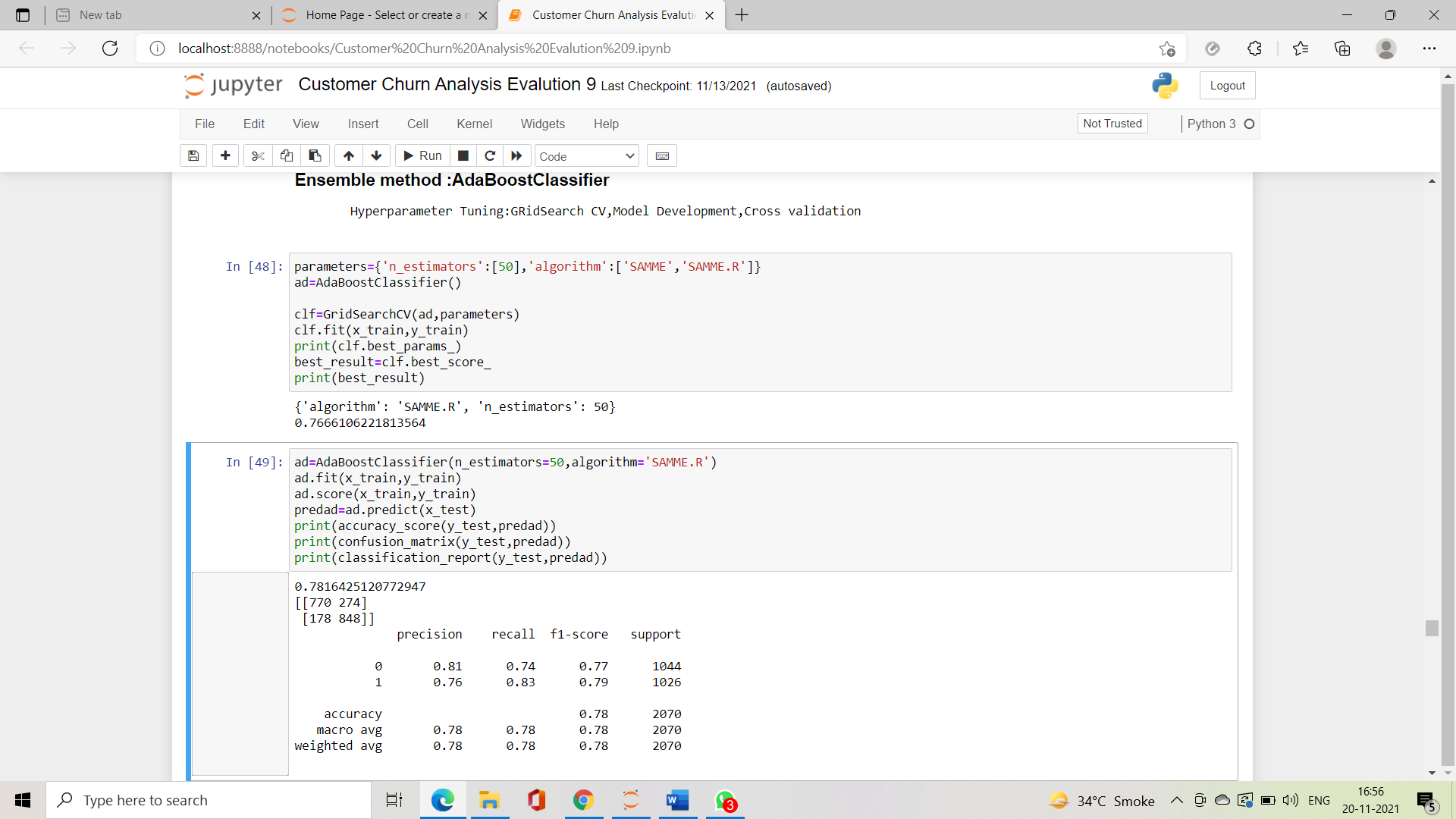


**RandomForestClassifier() -Model 2nd Result**

**Hyperparameter Tuning:GRidSearch CV-88,Model Development-91**

**Cross validation-90**

**Ensemble method :AdaBoostClassifier 3rd Model**

**Hyperparameter Tuning:GRidSearch CV-76,Model Development-78,Cross validation-7**

**Random Forest Classifier()**

**since cross validation score accuracy score are almost same.we are heading with a good approach.**

**AUC ROC Curve-**

The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.

The receiver operating characteristic, or ROC, curve is a popular plot for simultaneously displaying the tradeoff between the true positive rate and the false positive rate for a binary classifier at different classification thresholds.

**auc=roc\_auc\_score(y\_test,predrf)**

**fpr,tpr,thresholds=roc\_curve(y\_test,predrf)**

**plt.plot(fpr,tpr,color='orange',label='ROC')**

**plt.plot([0,1],[0,1],color='darkblue',linestyle='--',label='ROC Curve(Area=%0.3f)'%auc)**

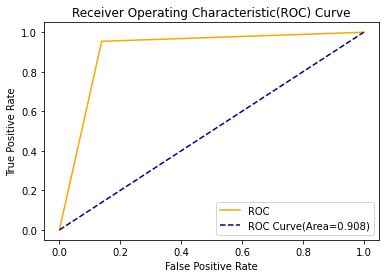
**plt.xlabel("False Positive Rate")**

**plt.ylabel('True Positive Rate')**

**plt.title("Receiver Operating Characteristic(ROC) Curve")**

**plt.legend()**

**plt.show()**

****

rf=RandomForestClassifier()

we are getting rf model accuracy 91 Cross validation Score 90 Hyperparameter tuning Grid Search CV Score 88 which is best.

so our model performing extremely well.

since cross validation score accuracy score are almost same.we are heading with a good approach.

**Saving Best Model-**

We further proceed to test the object that we saved using joblib or pickle, and create a dataframe of predicted values.

import pickle

filename='Customer\_Churn\_Analysis\_IBM.pkl'

pickle.dump(rf,open(filename,'wb'))

**Conclusion:**

Following are the results that we achieve, with an accuracy of 91%.

import numpy as np

a=np.array(y\_test)

predicted=np.array(rf.predict(x\_test))

dfs=pd.DataFrame({'Original':a,'Predicted':predicted},index=range(len(a)))

dfs

| **Original** | **Predicted** |
| --- | --- |
| **0** | 1 | 1 |
| **1** | 0 | 0 |
| **2** | 1 | 1 |
| **3** | 1 | 1 |
| **4** | 1 | 1 |
| **...** | ... | ... |
| **2065** | 1 | 1 |
| **2066** | 0 | 0 |
| **2067** | 0 | 0 |
| **2068** | 0 | 0 |
| **2069** | 1 | 1 |

2070 rows × 2 columns

This marks the end of our process.

we have successfully trained our model to predict the customer data from IBM Sample Data Sets with the aim of building and comparing several customer churn prediction models with an accuracy of ~91%

we can see our prediction comparision pairplot.almost we are very close predict to churn.

sns.pairplot(dfs)

