Detecting Botnet Attacks in IoT Environments: An Optimized Machine Learning Approach

In proposal and modules documents file already we mention about paper details and implementation details and now in this documents we are providing output screens with extension.

In this paper to train and test all algorithm performance author has utilized BOTNET2018 dataset which can be download from below link

<https://cloudstor.aarnet.edu.au/plus/s/umT99TnxvbpkkoE?path=%2FCSV%2FTraning%20and%20Testing%20Tets%20(5%25%20of%20the%20entier%20dataset)%2FAll%20features>

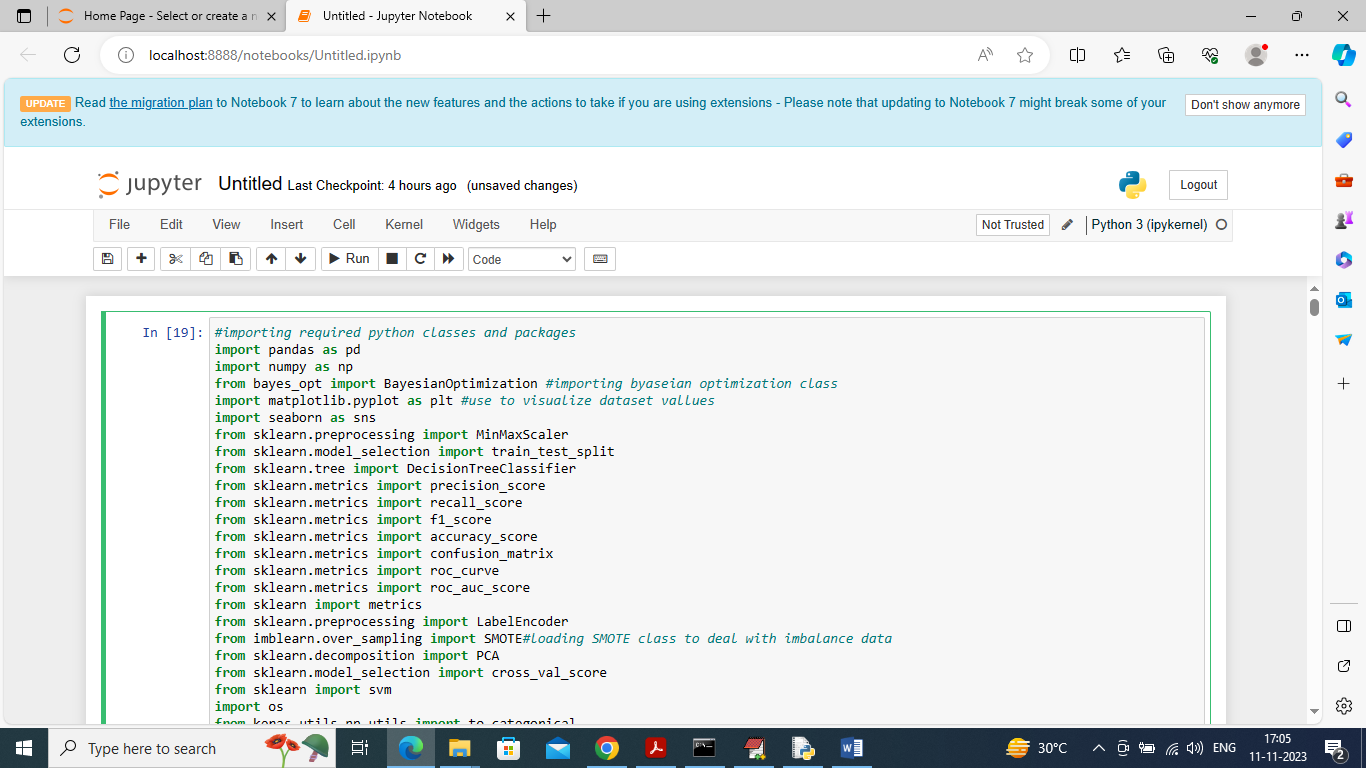
In propose work author has used Default Decision Tree and SVM algorithms as the existing algorithm and then BOGP Optimized Decision tree is the propose algorithm.

Extension Concept

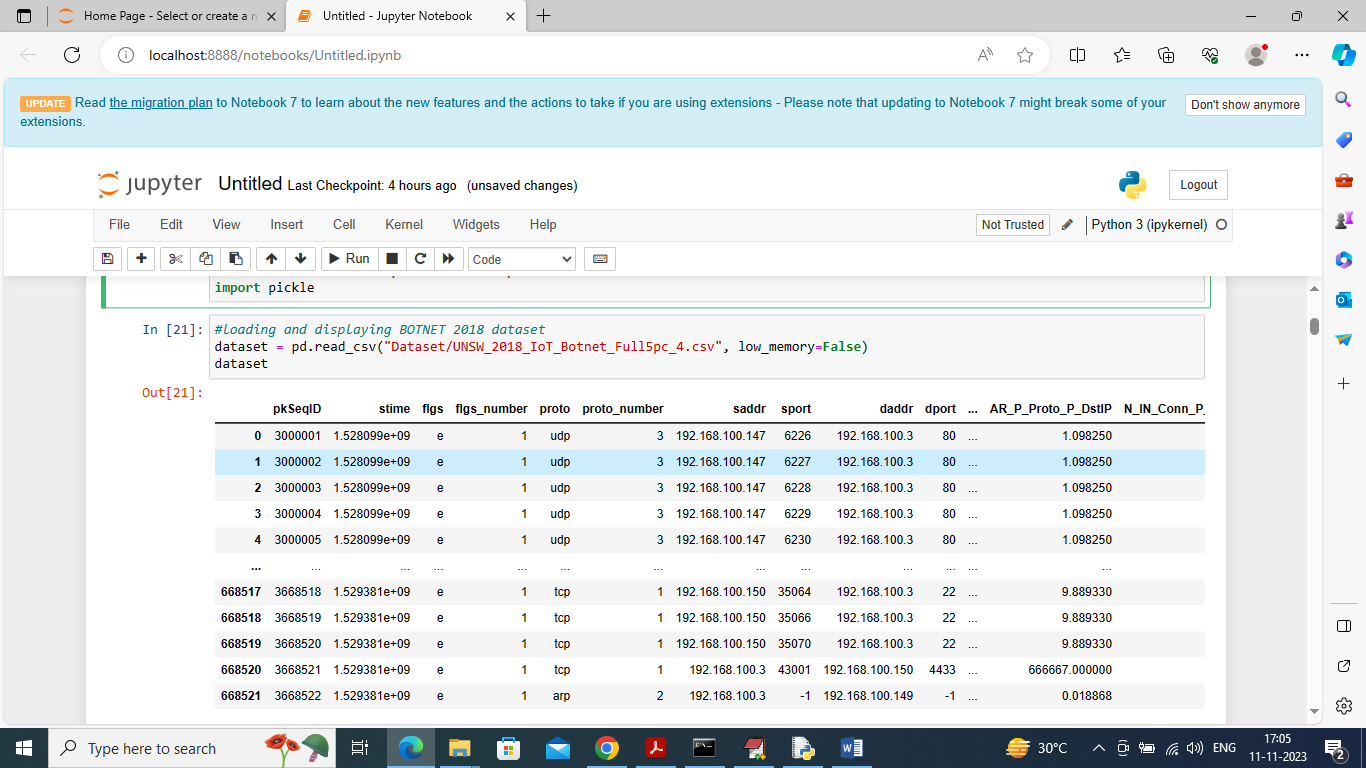
In propose paper author has used all traditional Machine Learning algorithms such as SVM and Decision Tree so as extension we have applied advance deep learning algorithm CNN (Convolution Neural Networks) which will optimized dataset features using multiple layers and neurons and can able to select more optimized features which can help in better prediction accuracy.

Output SCREEN SHOTS

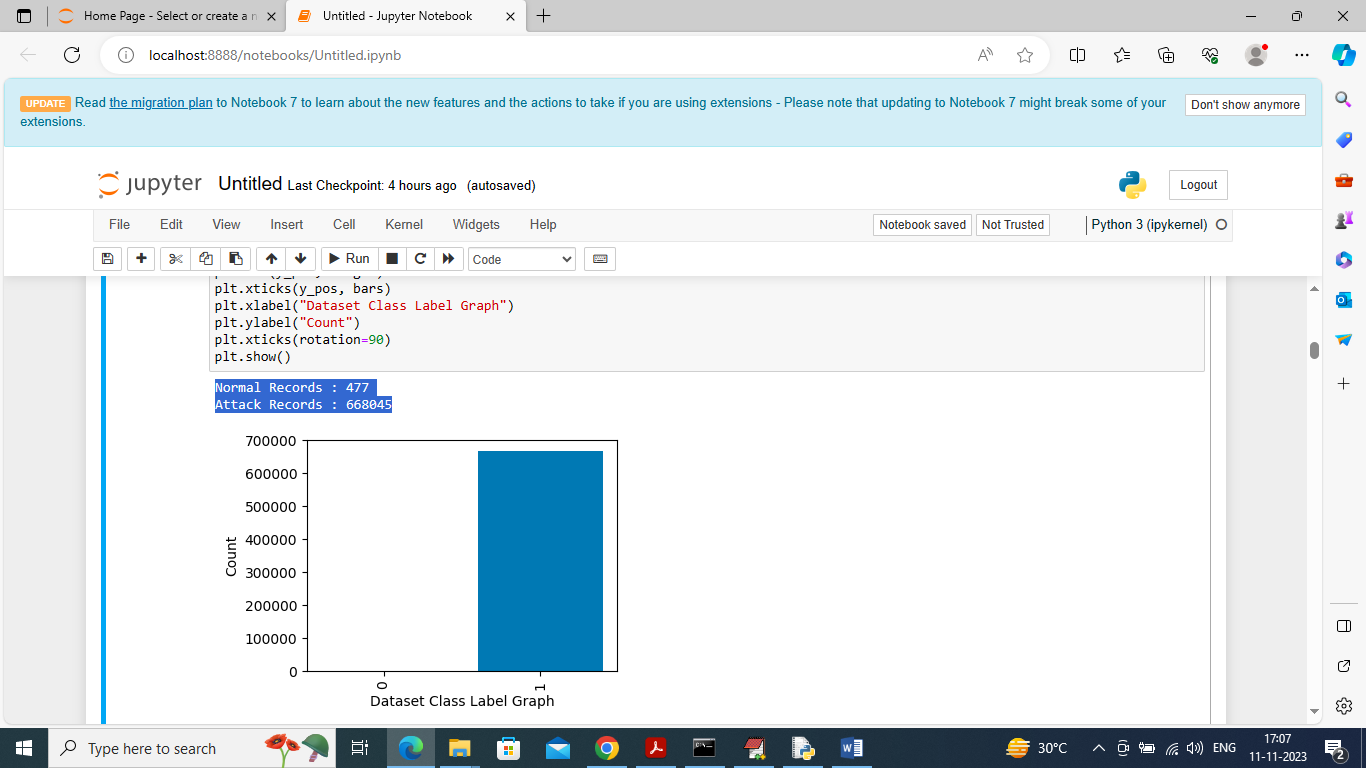
We have coded this project using JUPYTER notebook and below are the code and output screen with blue colour comments



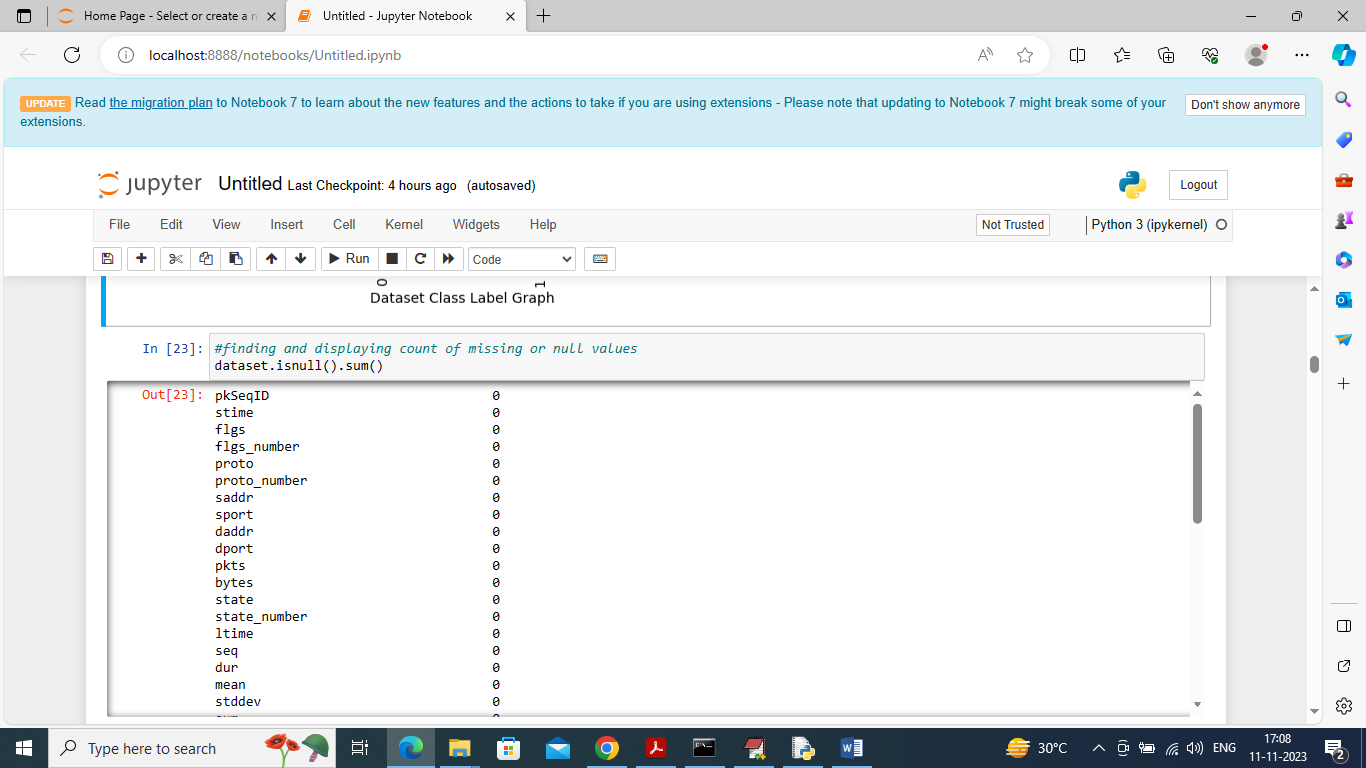
In above screen importing required python classes and packages



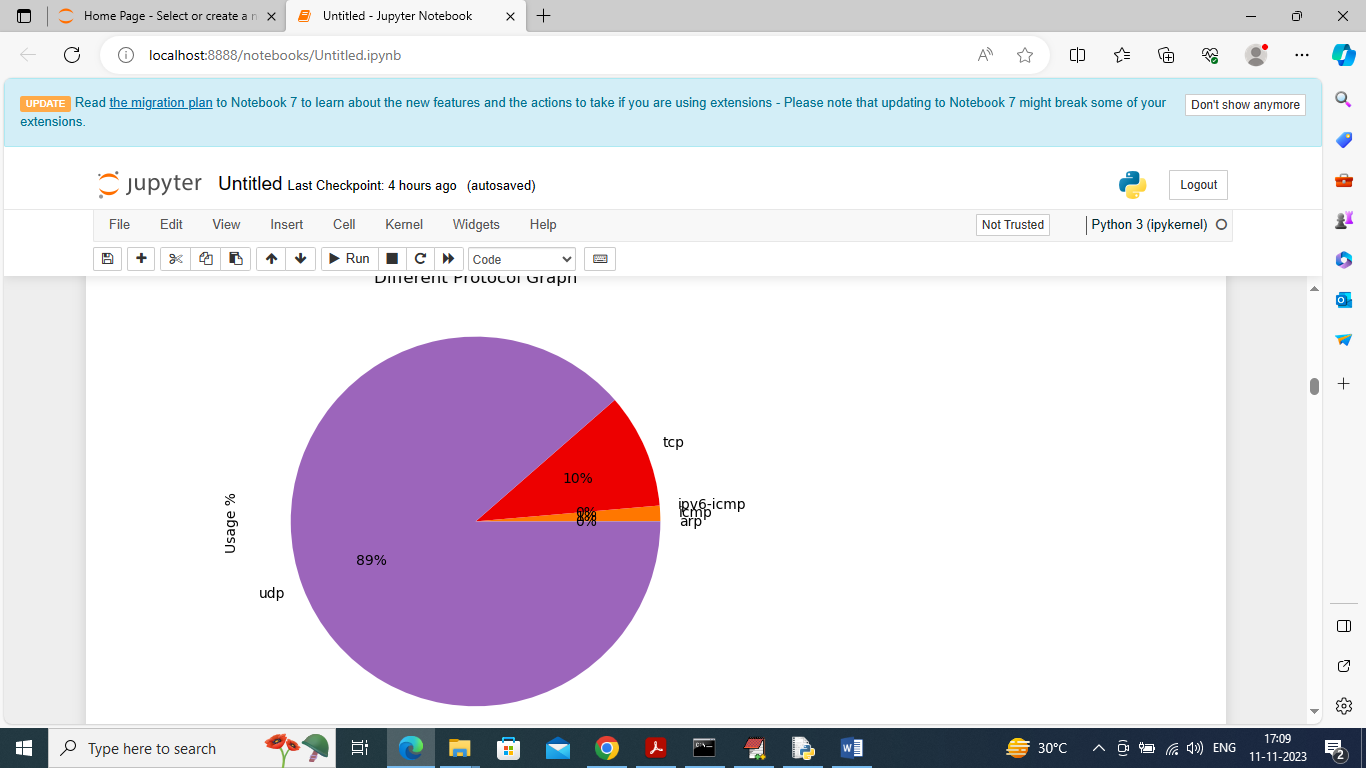
In above screen loading and displaying BOTNET2018 dataset values and in above dataset we have both numeric and non-numeric values but ML algorithms take only numeric data so later by applying processing techniques will convert non-numeric values to numeric values



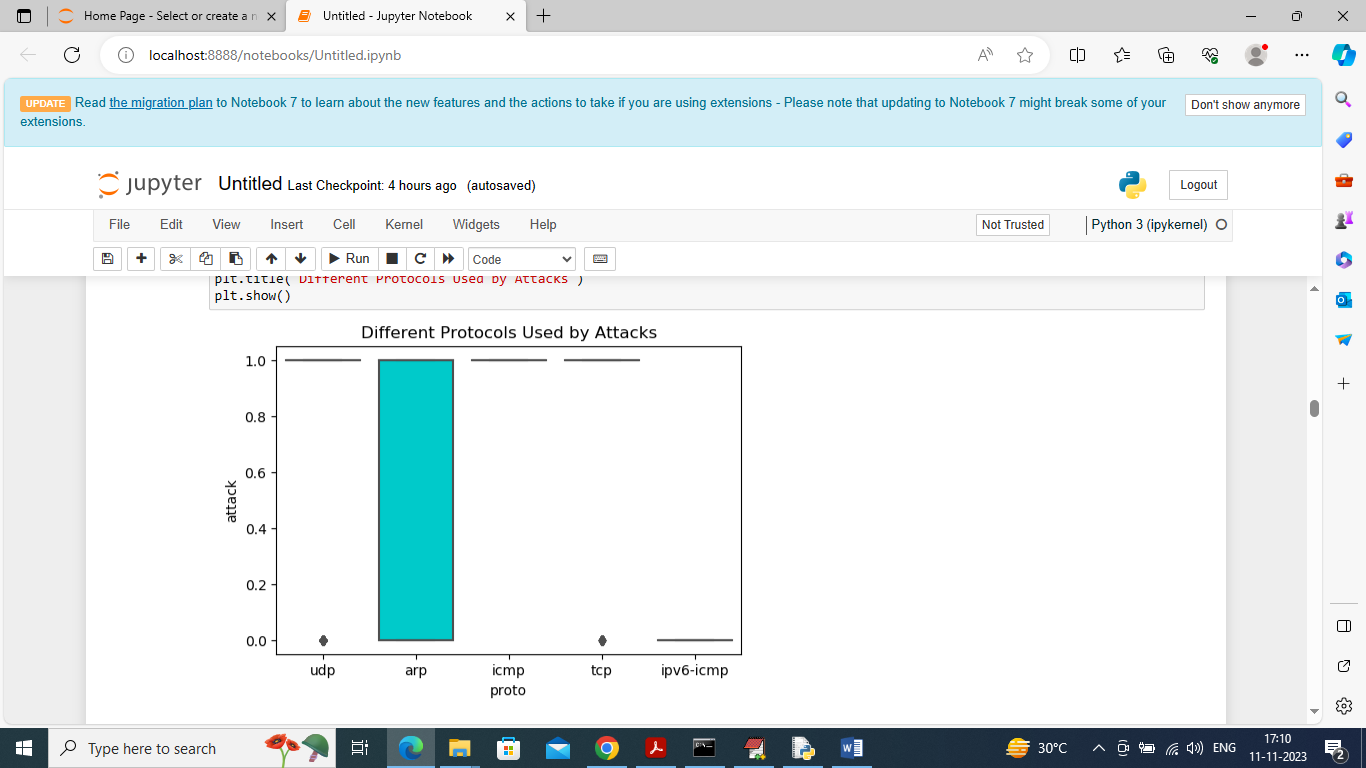
In above screen displaying number of NORMAL and ATTACK records found in dataset and in above blue colour text can see Normal contains only 477 records and attack contains records in lakhs and in graph also we can see only attack records are more so this dataset is highly imbalance and we can balance by applying SMOTE technique



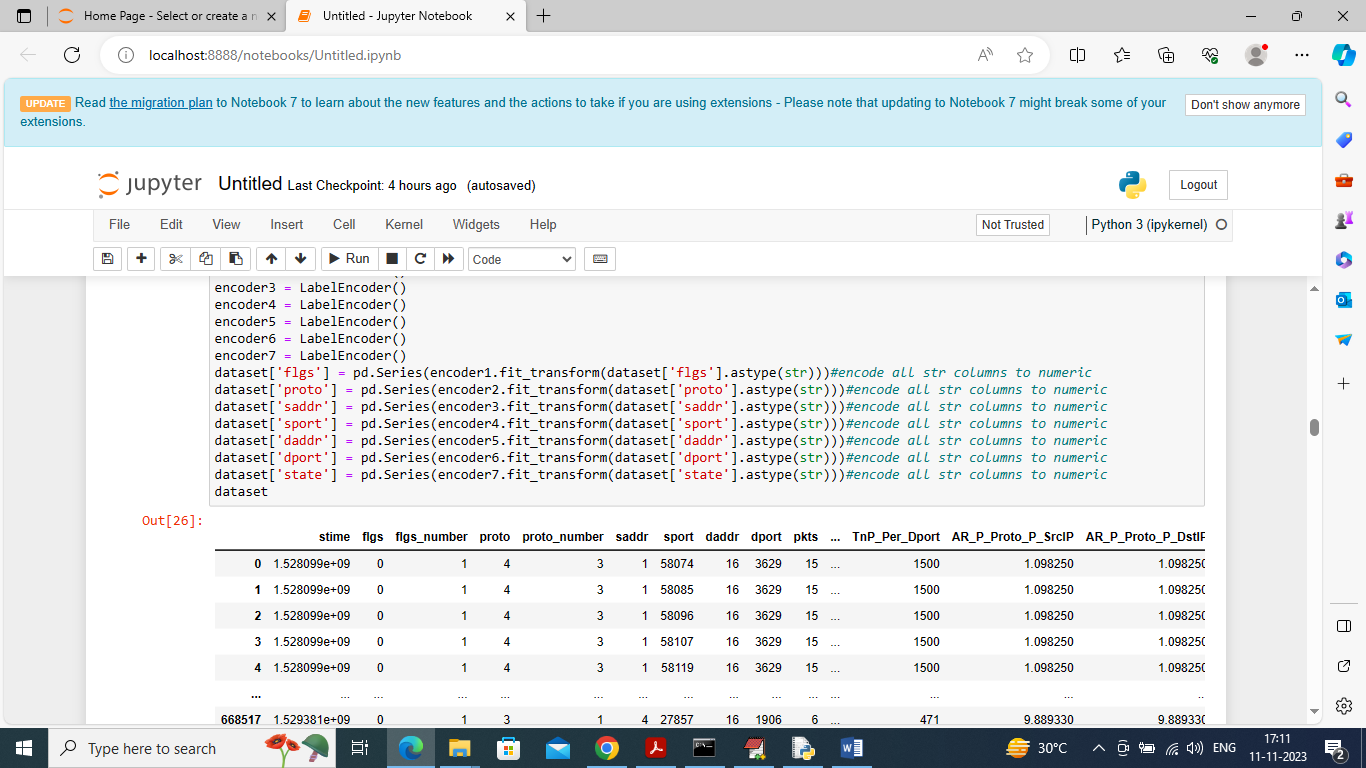
In above screen finding and checking count of missing values but this dataset contains NO missing values



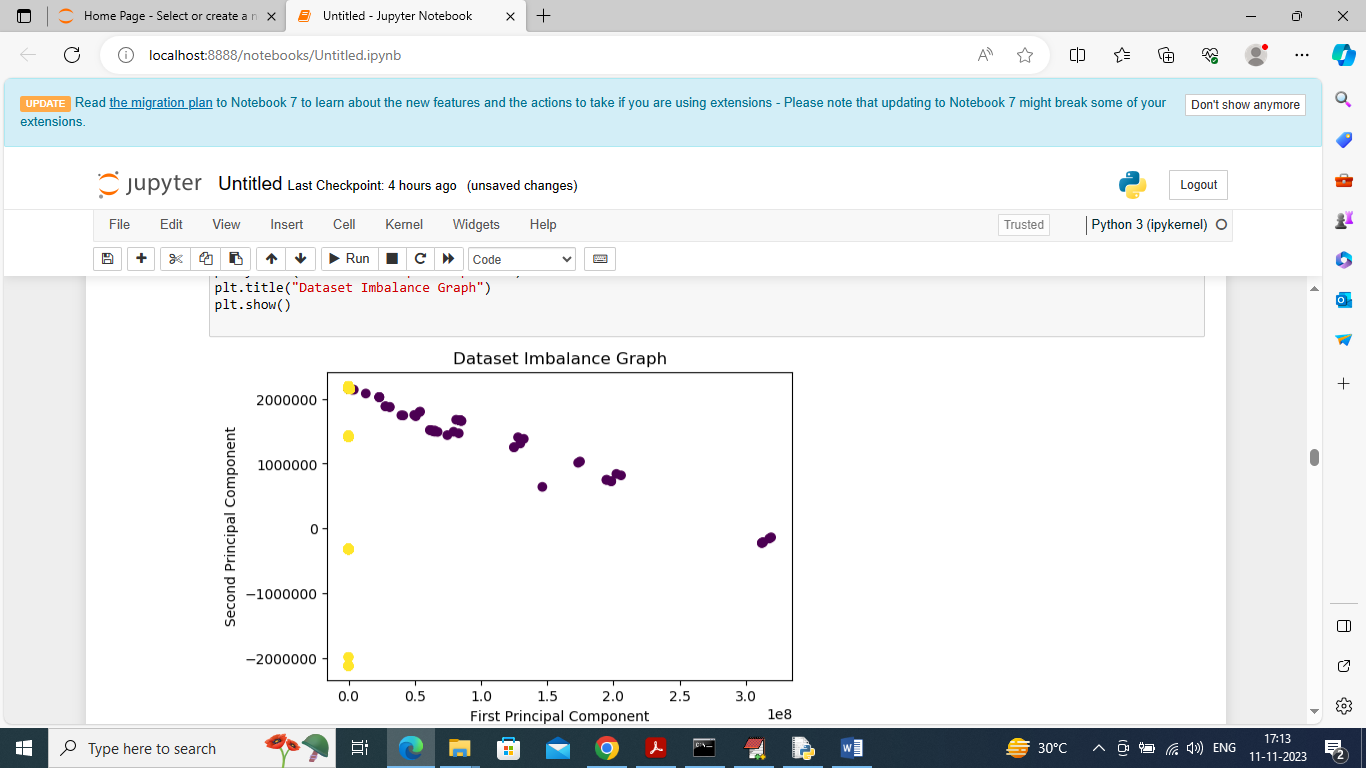
In above graph displaying percentage of different protocols used in this dataset and from above graph can say 89% is usage of UDP protocol



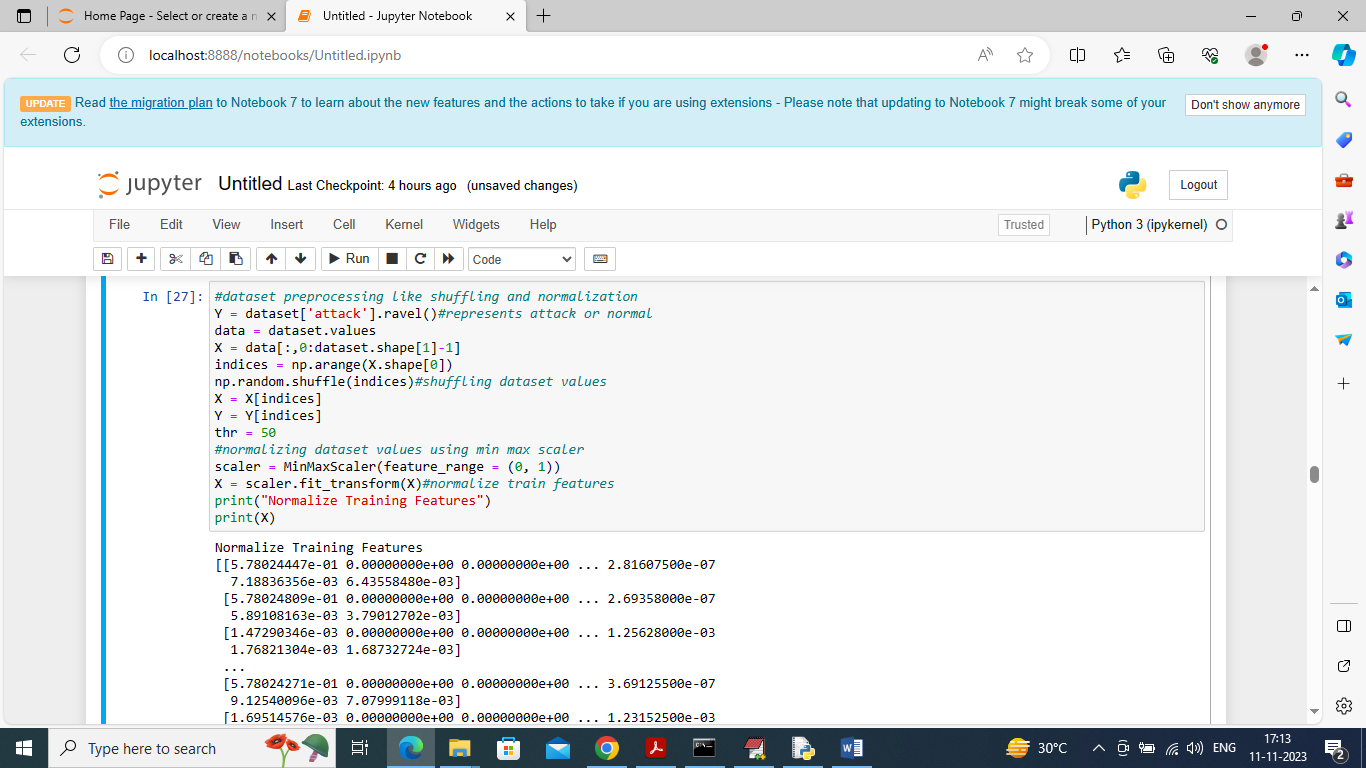
In above graph finding different protocols uses by attacks and in above graph x-axis represents Protocol Type and y-axis represents usage %



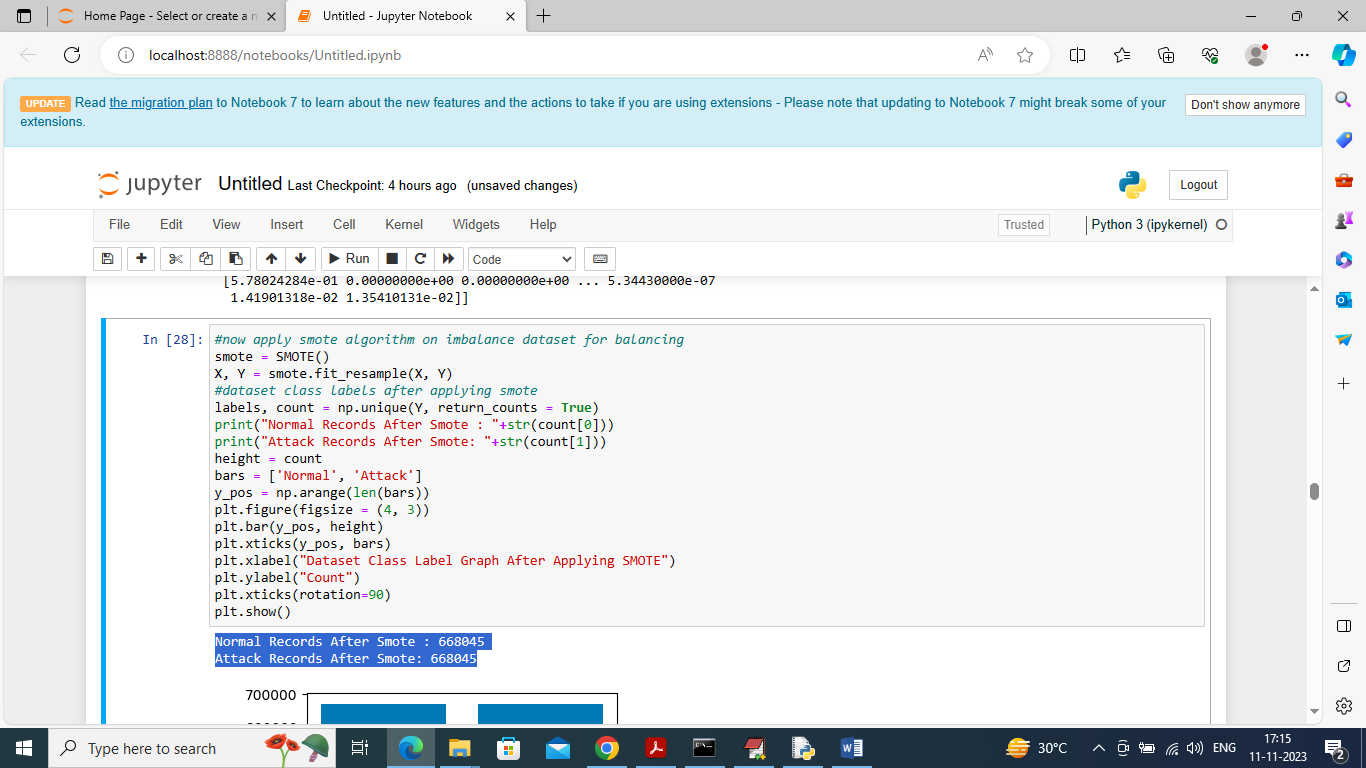
In above screen by applying Label Encoder class we are converting non-numeric values into numeric values and then displaying all numeric data



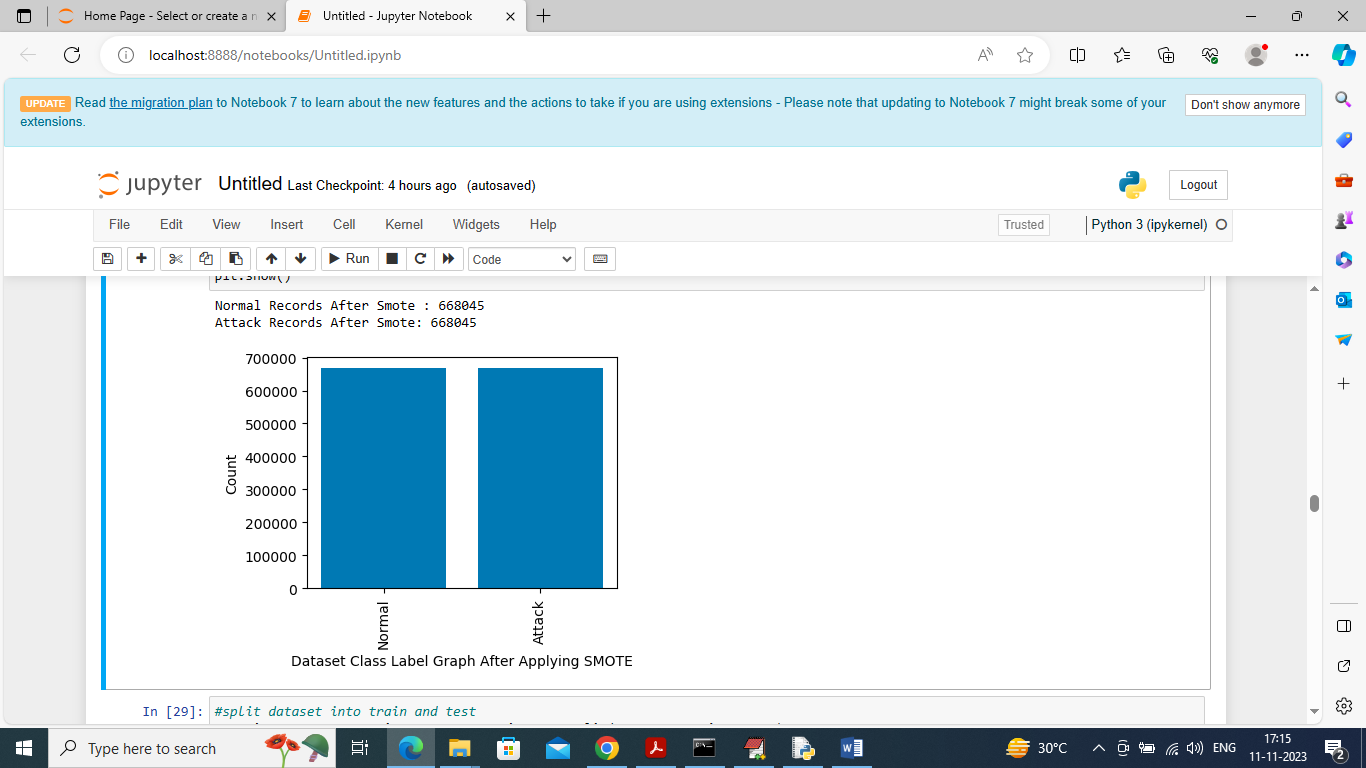
In above screen displaying dataset imbalance ratio where blue dots are for ATTACKS and yellow dots for Normal records and can see there are so many blue dots and yellow dots are very few so this dataset is highly imbalance



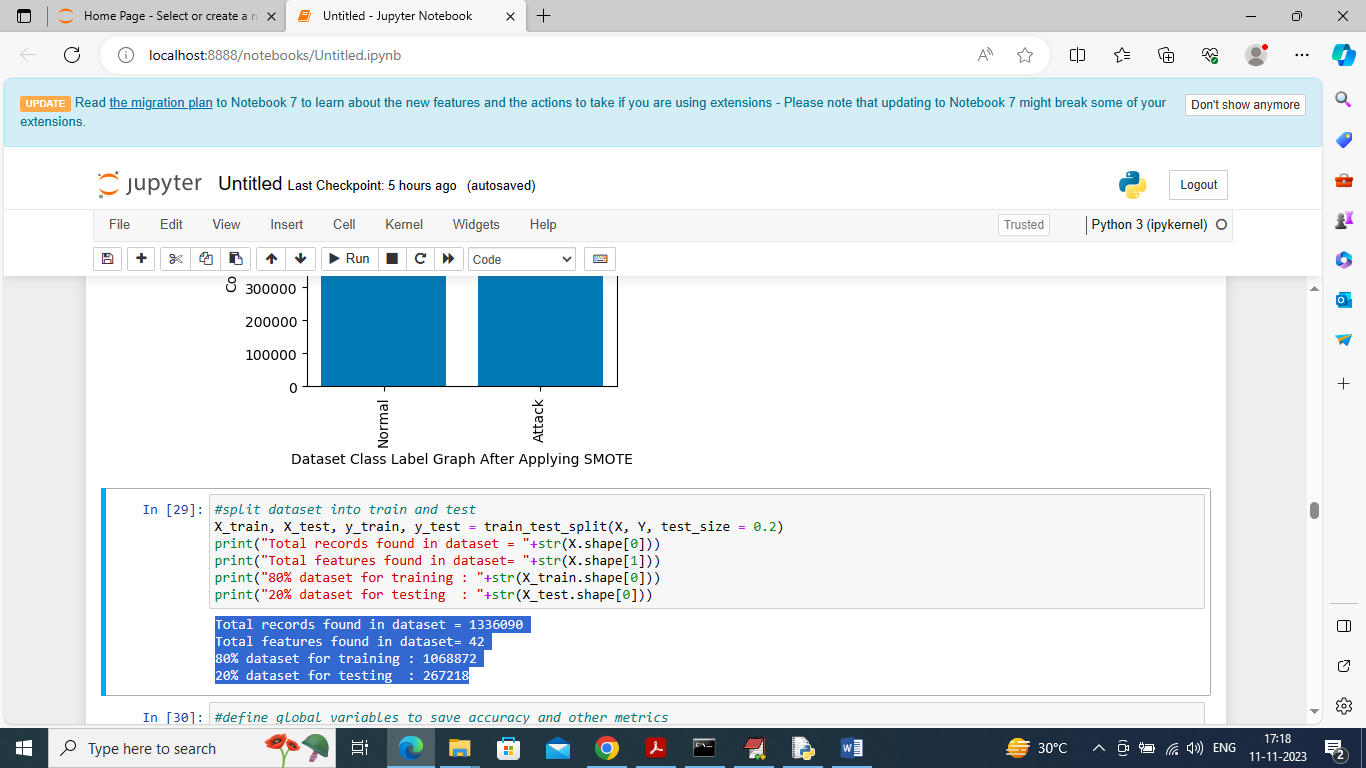
In above screen applying various pre-processing techniques such as shuffling and normalizing dataset using MINMAX scaling and then displaying normalized values



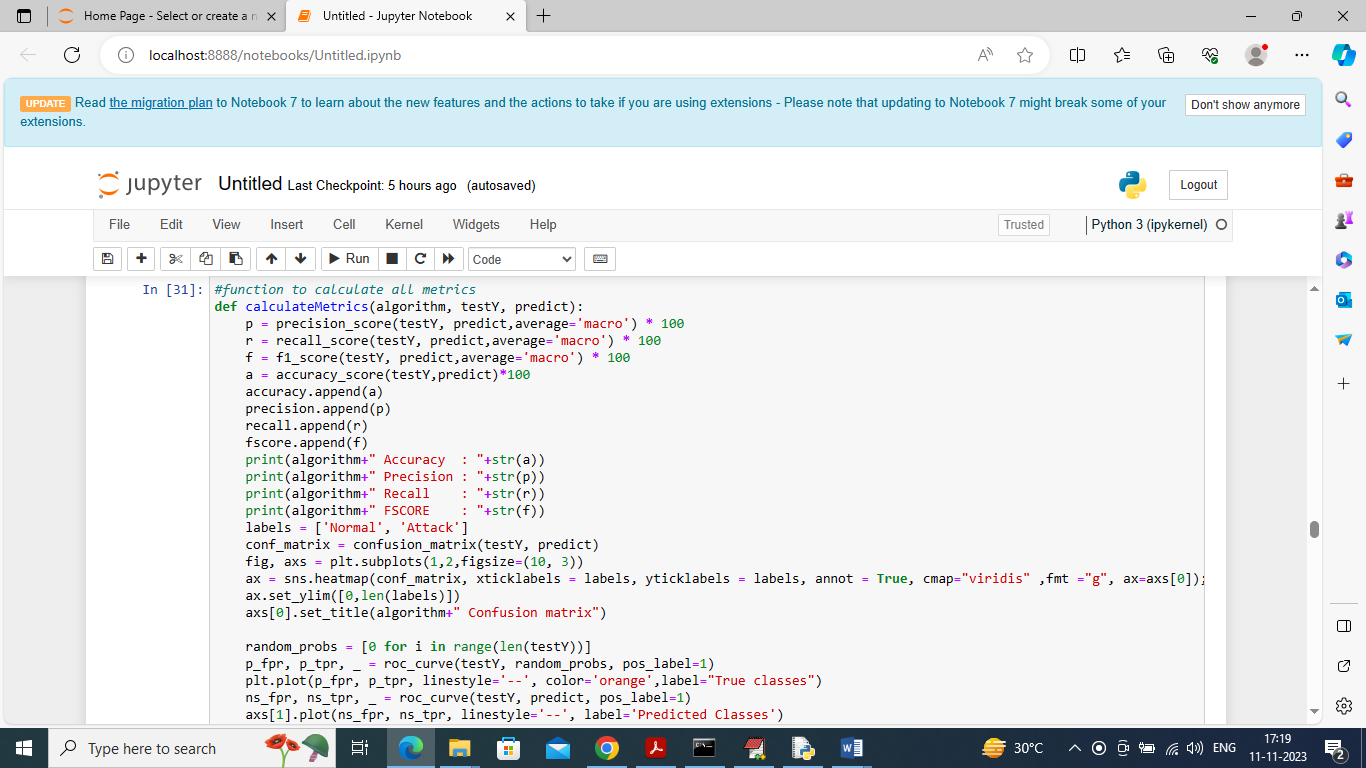
In above screen applying SMOTE algorithm on processed dataset and in blue colour text can see both normal and attacks records size are equal and now dataset is balance



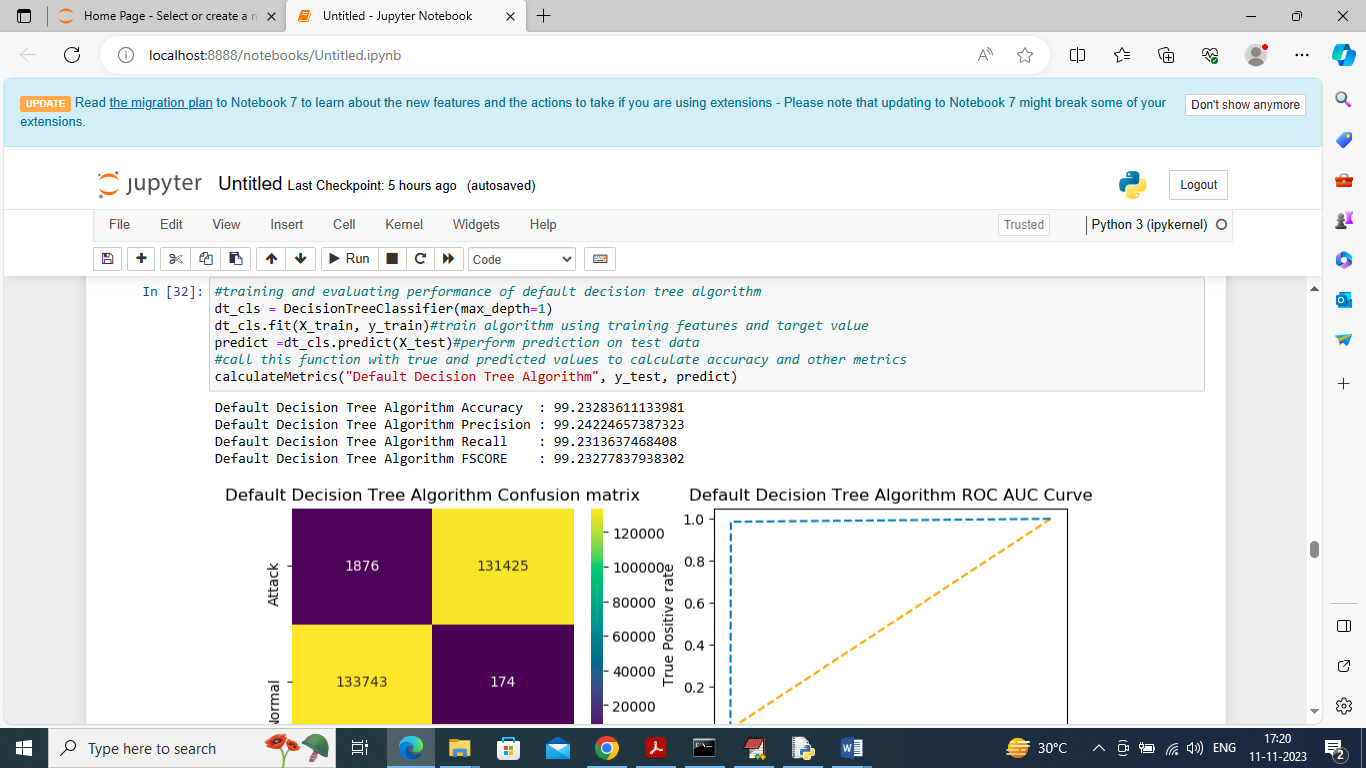
After applying SMOTE in above graph can see both class labels are having equal number of records



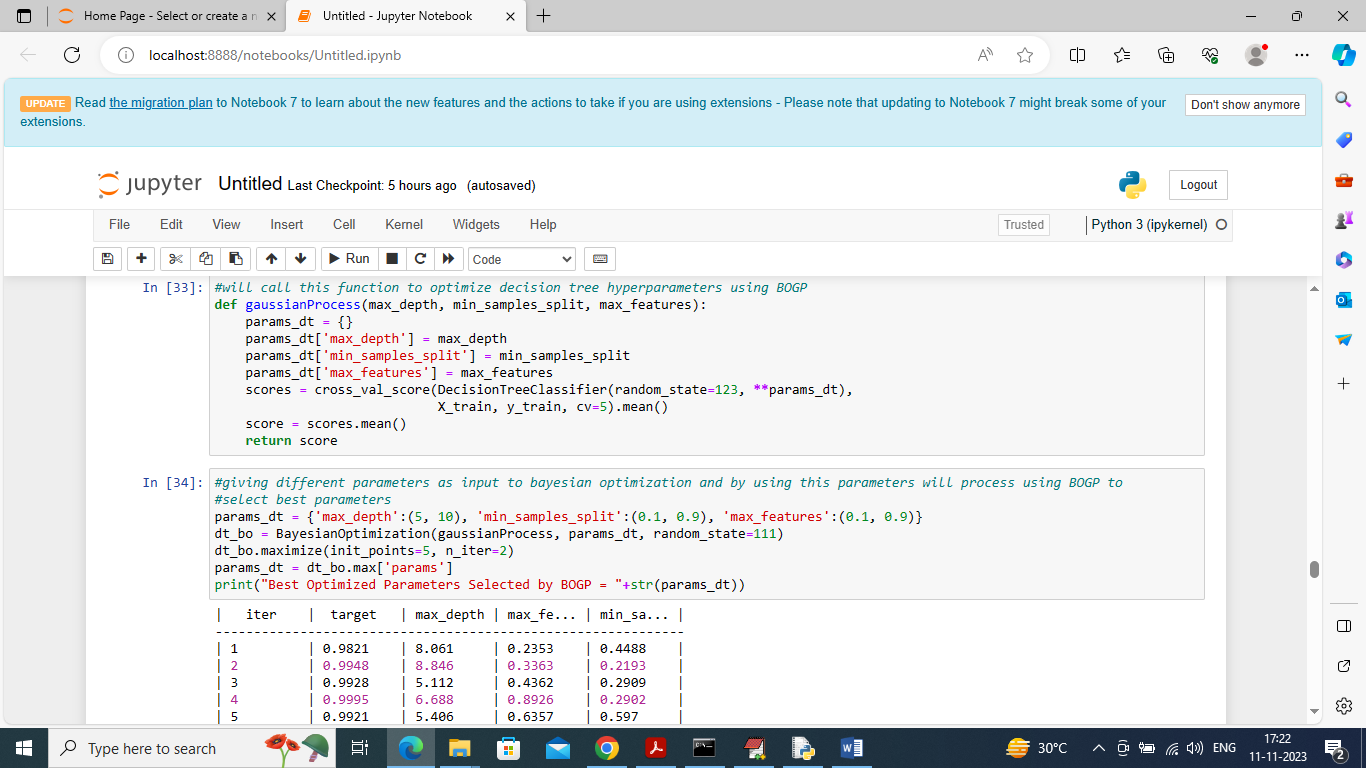
In above screen splitting dataset into train and test where application using 80% dataset for training and 20% for testing



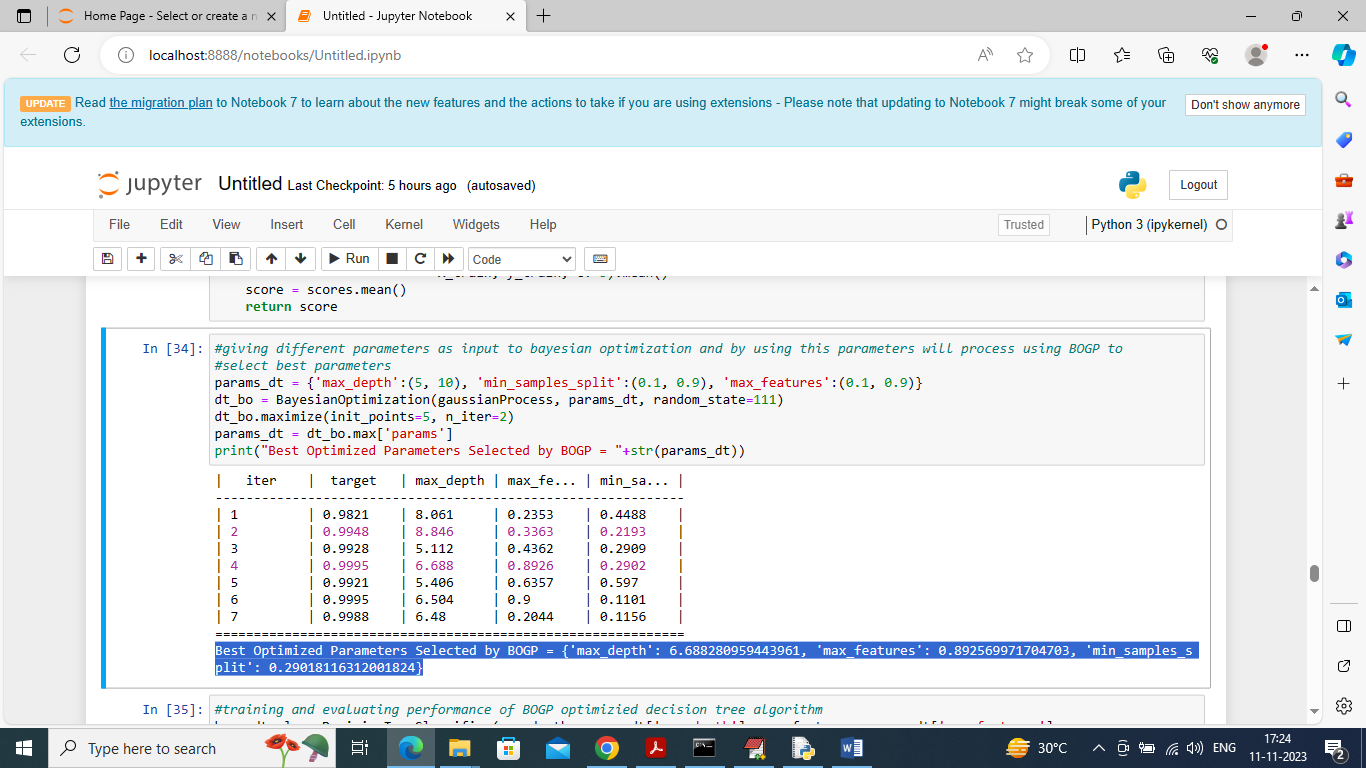
In above screen defining function to calculate accuracy and other metrics



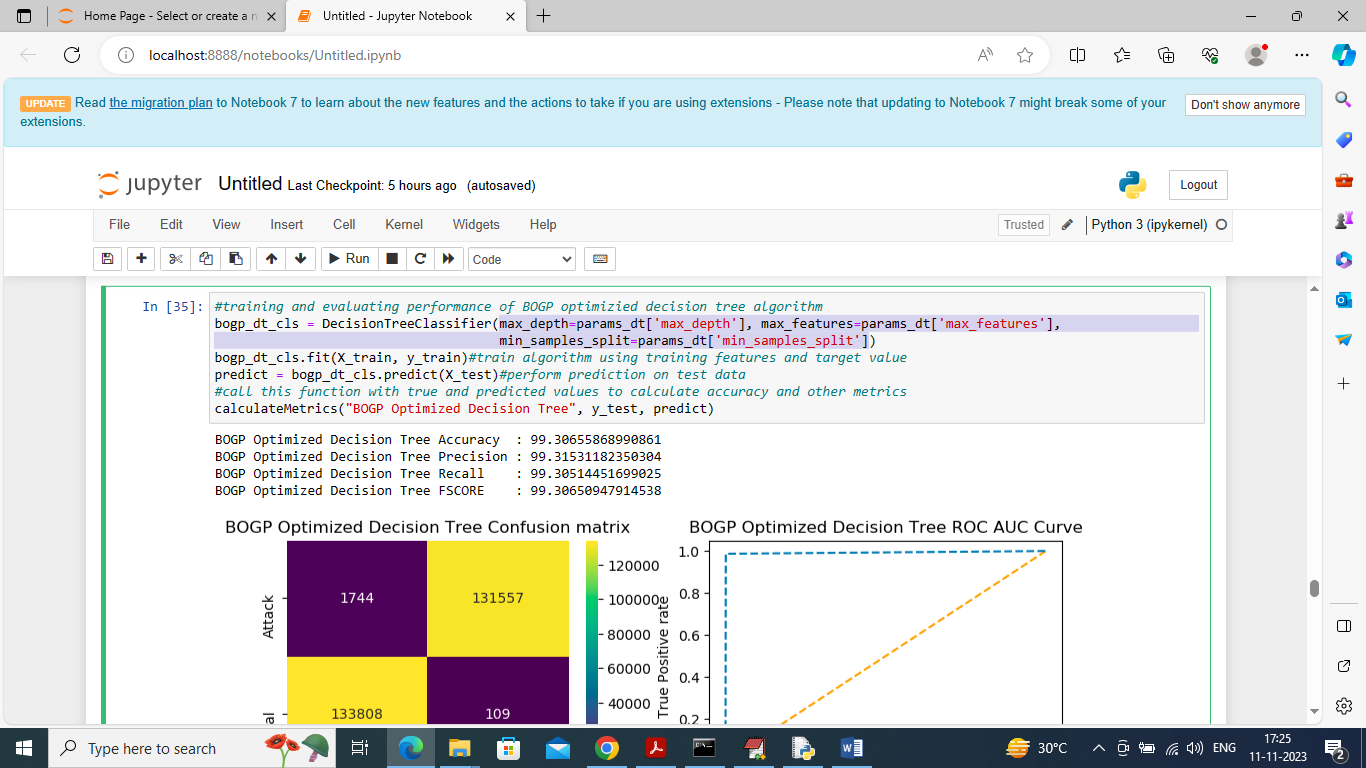
In above screen training Default Decision Tree algorithm and its 99.23% as the accuracy and can see other metrics also. In above confusion matrix graph x-axis represents “Predicted Labels” and y-axis represents “True Labels” and all different yellow boxes in diagnol represents correct prediction count and remaining all blue boxes contains incorrect prediction count which are very few. In Roc curve graph x-axis represents False Positive Rate and y-axis represents True Positive Rate and if blue line goes below orange line then all predictions are incorrect and if goes above orange line then all predictions are correct and in above ROC graph we can see all predictions are correct up to 100%



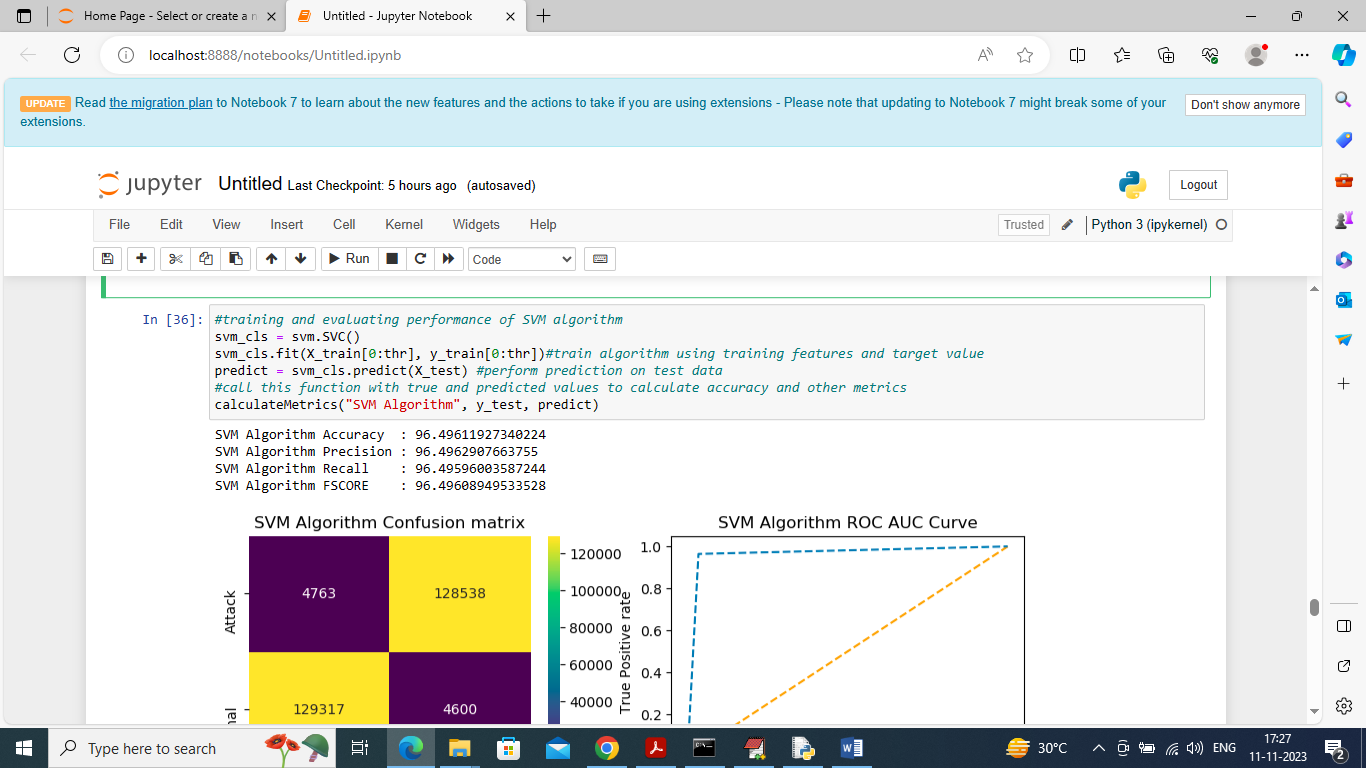
In above screen defining Gaussian Process and optimization function called BOGP with various parameters and then in output in tabular format we can see Target as the accuracy. BOGP will iterate with various parameters values and then trained and calculate accuracy and this will continue till no more enhancement left. In above table we can see for each iteration we can see Target accuracy and used Parameters values and in last will get below ‘Optimized Parameter’



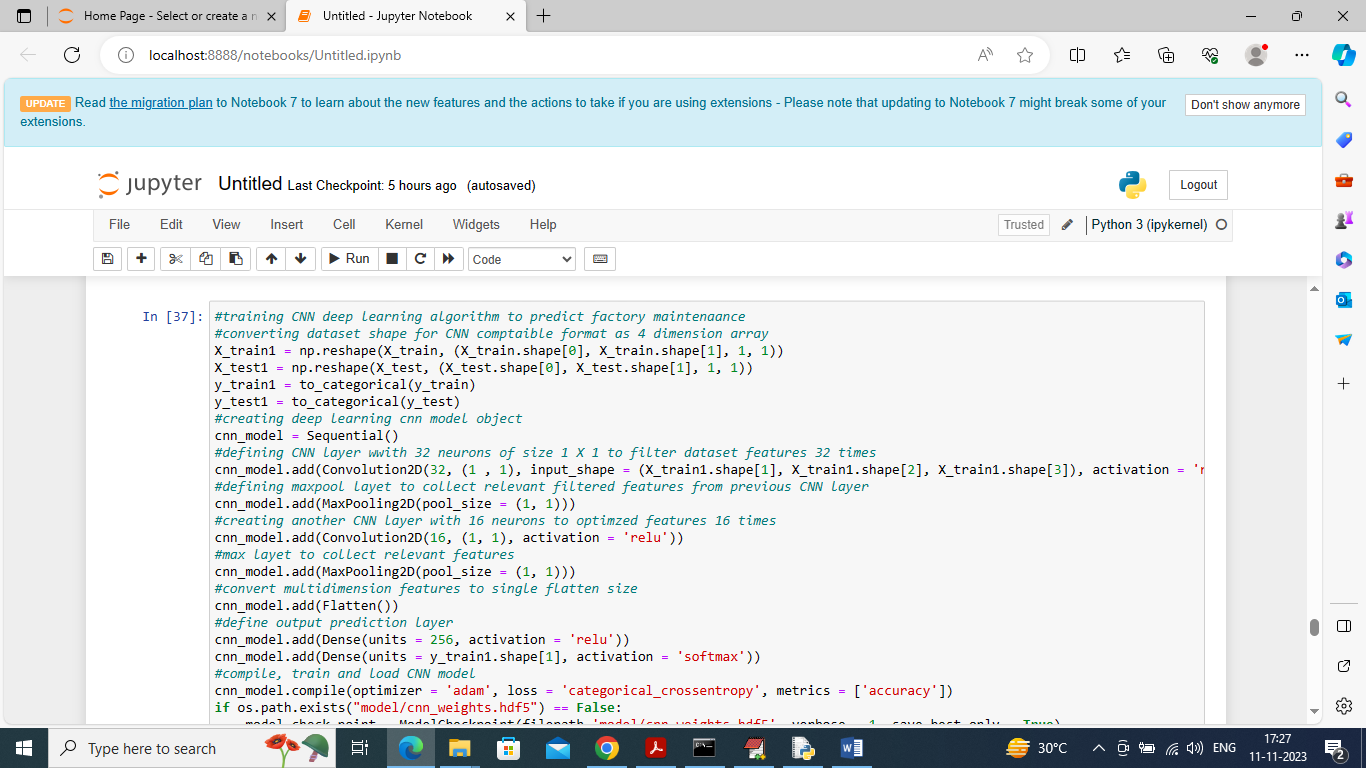
In above screen in blue colour text can see optimized parameters with highest Target values is selected as best parameters and now in optimized decision Tree will use above parameters



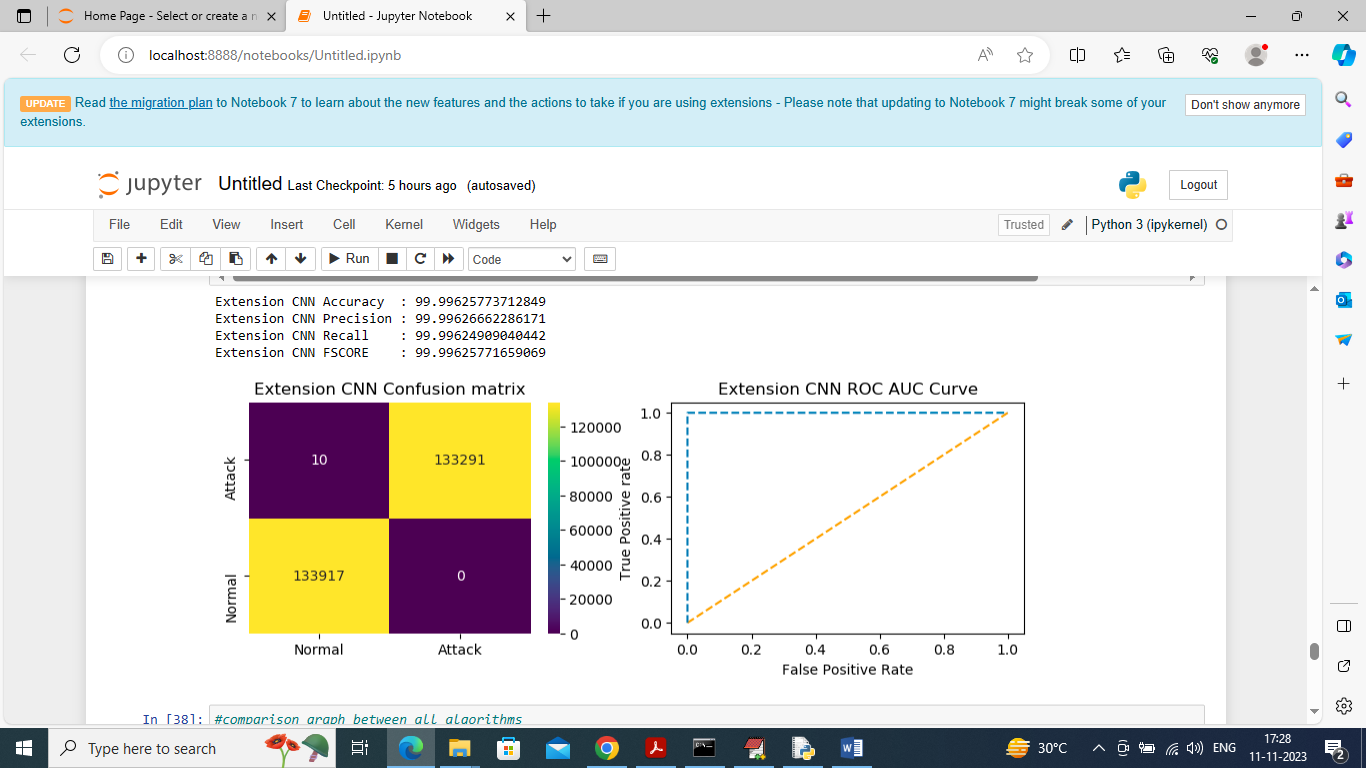
In above screen in light blue text can see that we are training decision tree with best selected BOGP parameters and after training optimized decision tree got 99.30% accuracy which is higher than default decision tree and can see other metrics values



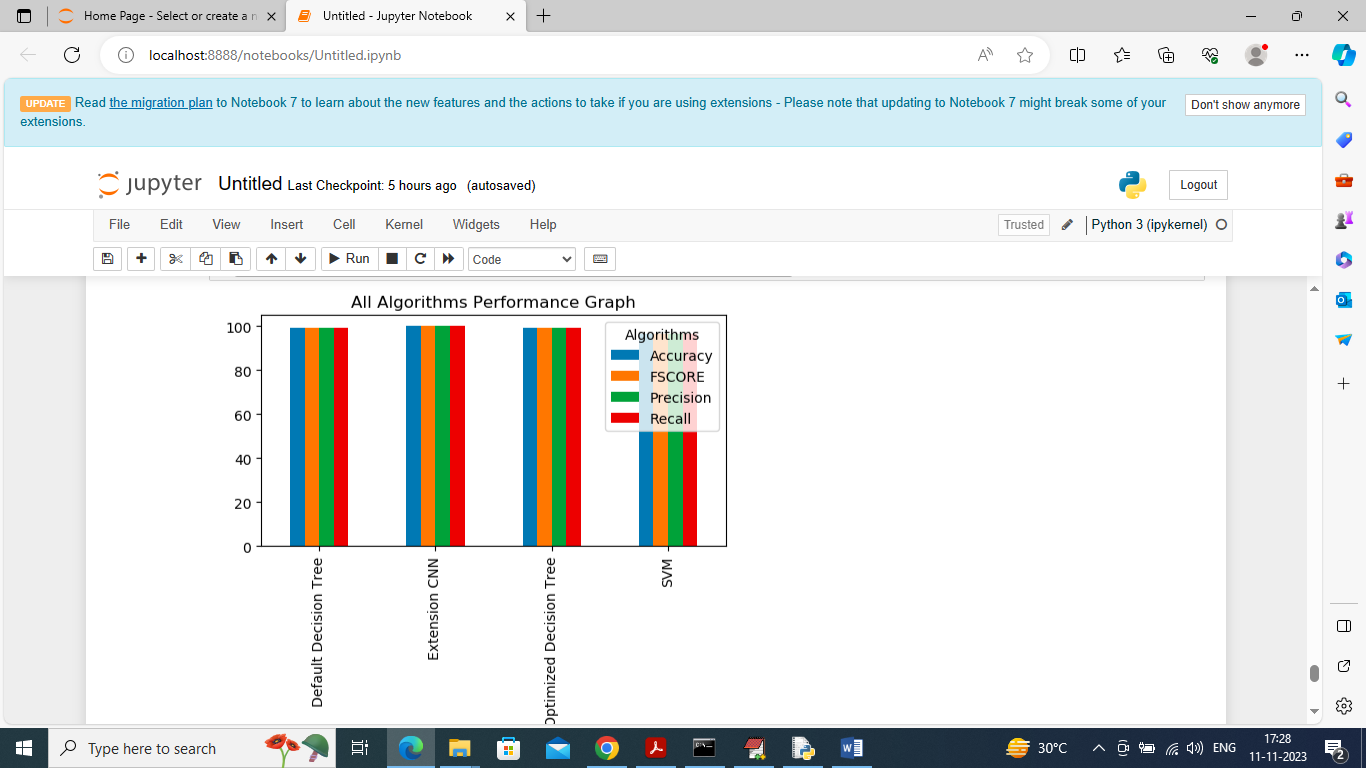
In above screen training existing SVM algorithm and it got 96% accuracy



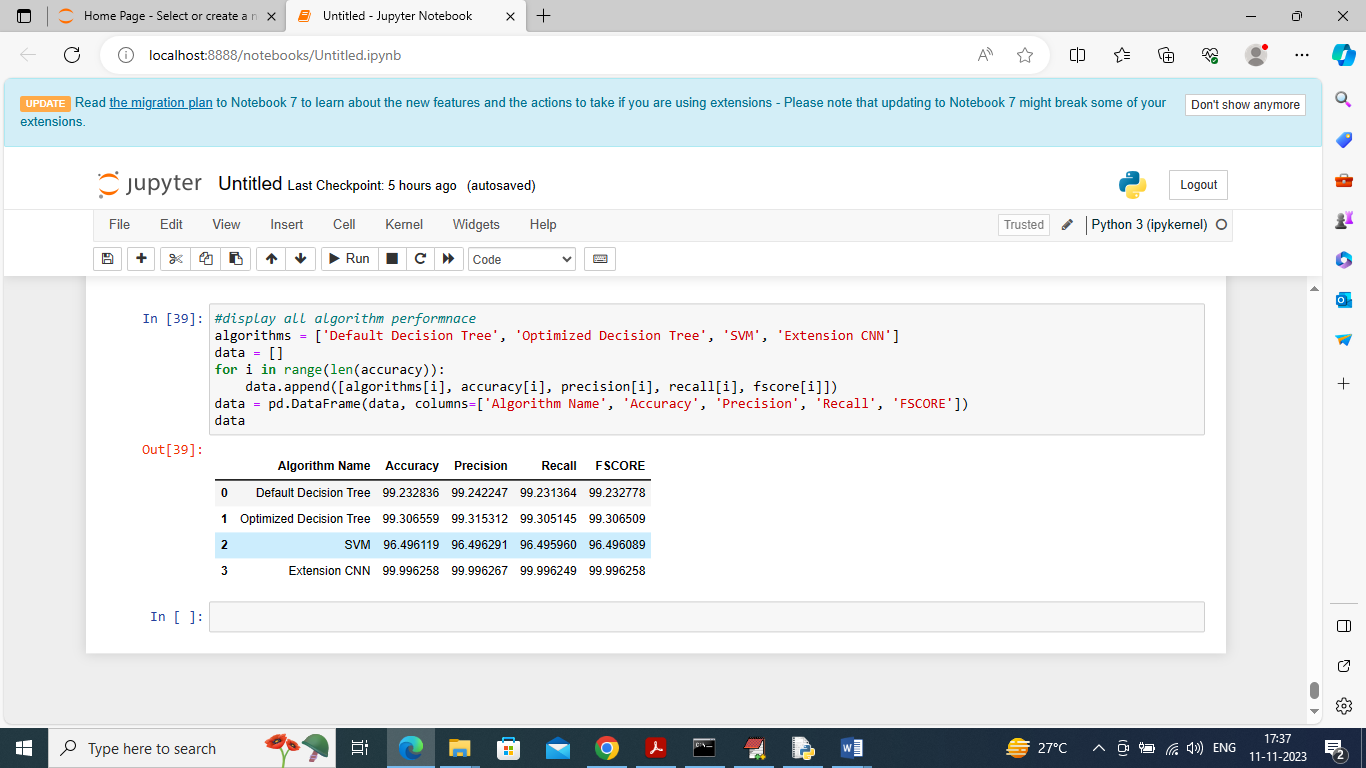
In above screen training deep learning extension CNN algorithm and after executing above block will get below output



In above screen extension CNN got 99.996% accuracy



In above graph displaying all algorithm performance where x-axis represents algorithm names and y-axis represents accuracy and other metrics in different color bars and in all algorithms extension CNN got high performance



In above screen displaying all algorithm performance in tabular format and in all algorithms can see extension CNN got high performance