**Building Classification Model on Insurance Claims- Fraud Detection**

Hello all, On this Article, I will cover the basic of creating your own classification Model using Python. I will try to explain and demonstrate to you step-by-step from preparing your data, training your model, optimising the model and how to save it for later use.

**Introduction**

In Machine learning, classification is the problem of identifying to which of a set of categories, a new observation belongs, based on a training set of data containing observations (or instances) whose category membership is known.

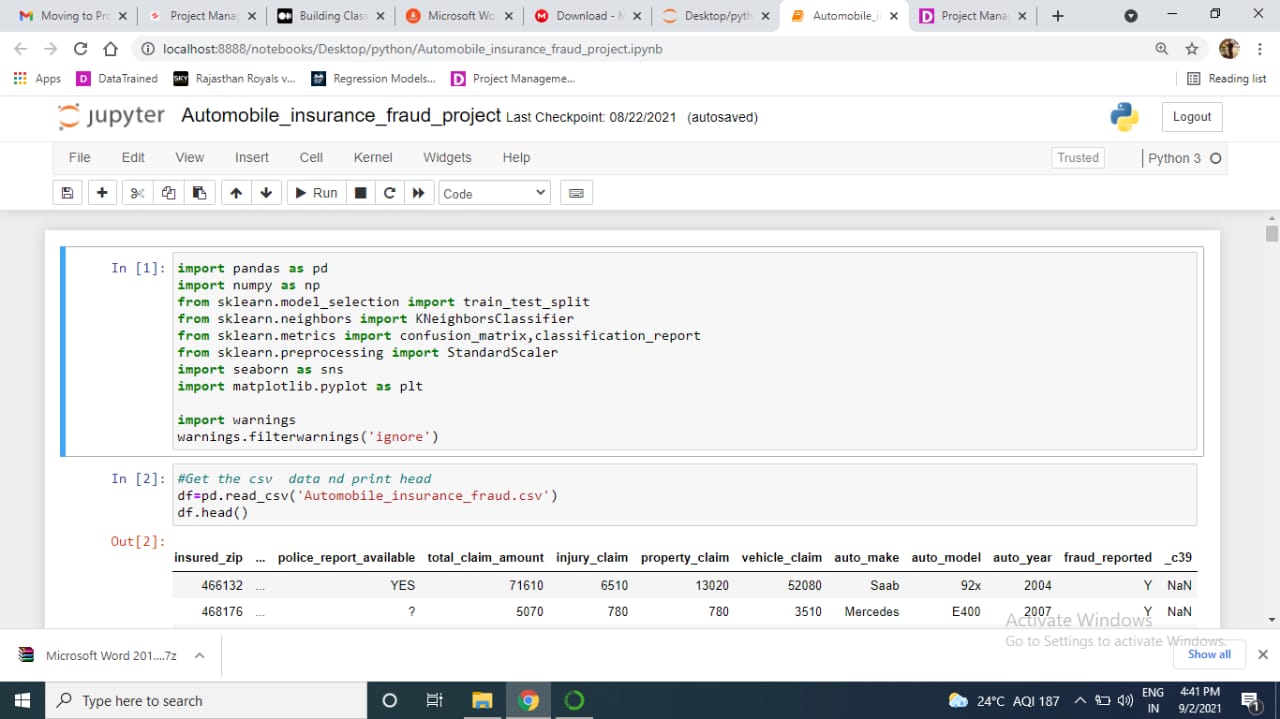
Some of the example of classification problems are:-

1. Assigning a disgnosis of a patient based on observed categories of the patient.
2. Deciding whether a received email is a spam or an organic-mail;

In this Article we will use the Automobile-insurance-fraud to build a model to predict whether someone is doing a fraud claims or not depending on some attributes. We will try to build a model with good accuracy using different-different algorithm i.e. DecisionTreeClassifier, LogisticRegression, K-NeighborsClassifier, and RandomForestClassifier. At the end we will evaluate and compare all the algorithm and select which one is best for our case. We will also try to optimise our model by tunning the HyperParameters of the model by using GridSearchCV.

Lastly, we will save the prediction result from our dataset and then save our model for re-usability.

Our first step is to load some basic libraries such as Pandas and NumPy etc.



**Data Pre-Processing**

Before we begin to create our first model, we first need to load and pre process. This step ensure that our model will receive a good data to learn from, as they said, “A Model is only as good as it’s data”.

The data preprocessing will be divided into few steps as explained below:-

1. **Loading Data**

In this first step, we will load our dataset.

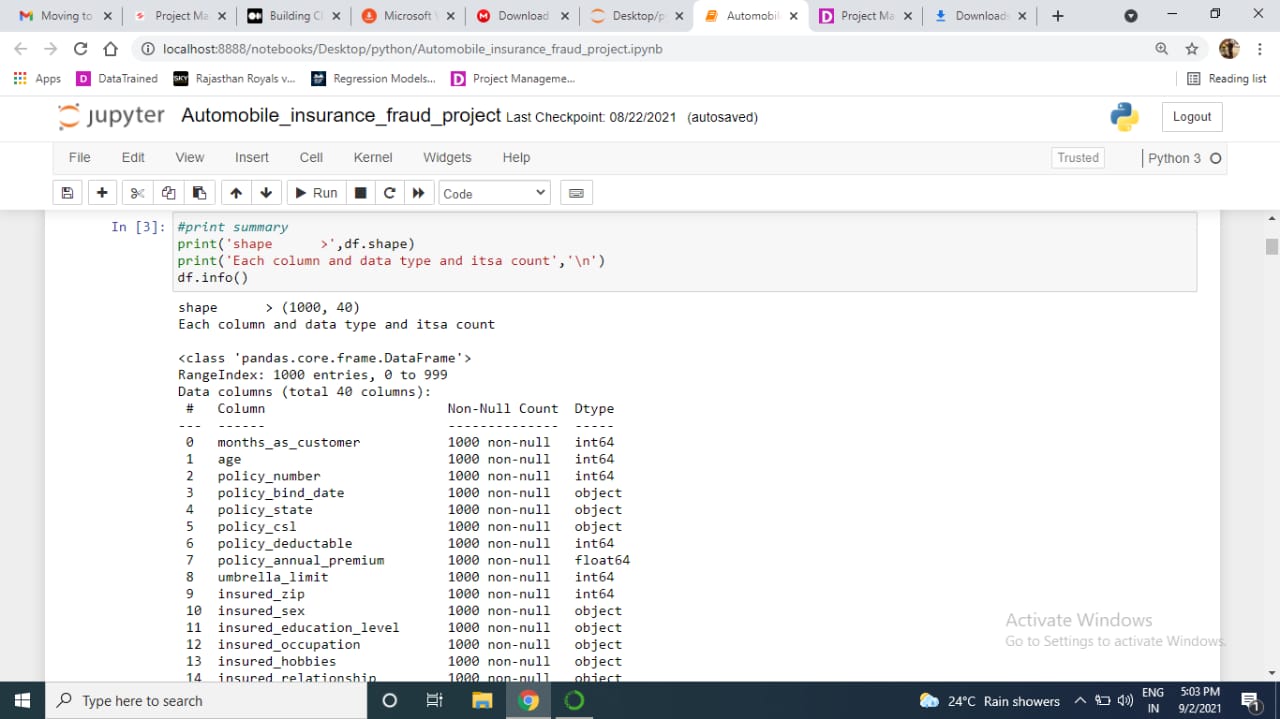
Input Variables:

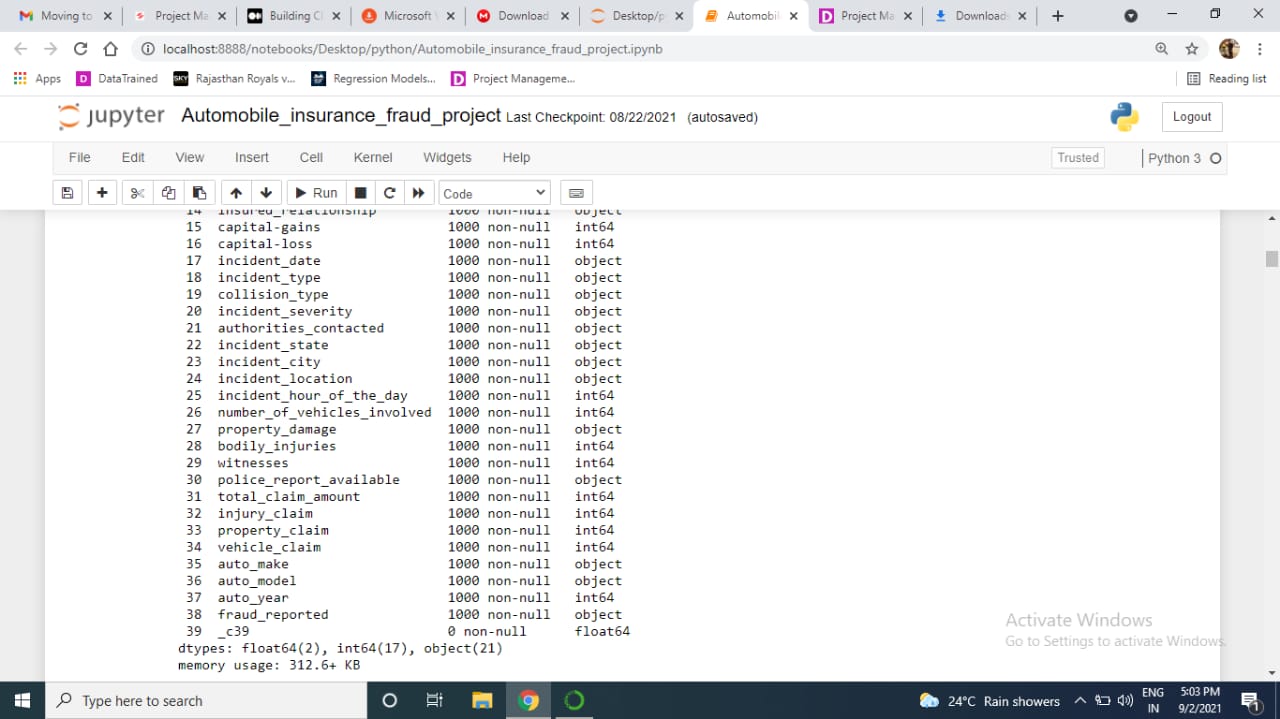
* There are total 40 columns and 1000 rows.
* Out of 39 columns, 18 columns are of object datatype.

Output Variables: (label)

‘fraud\_reported’(binary:’Y’,’N’)

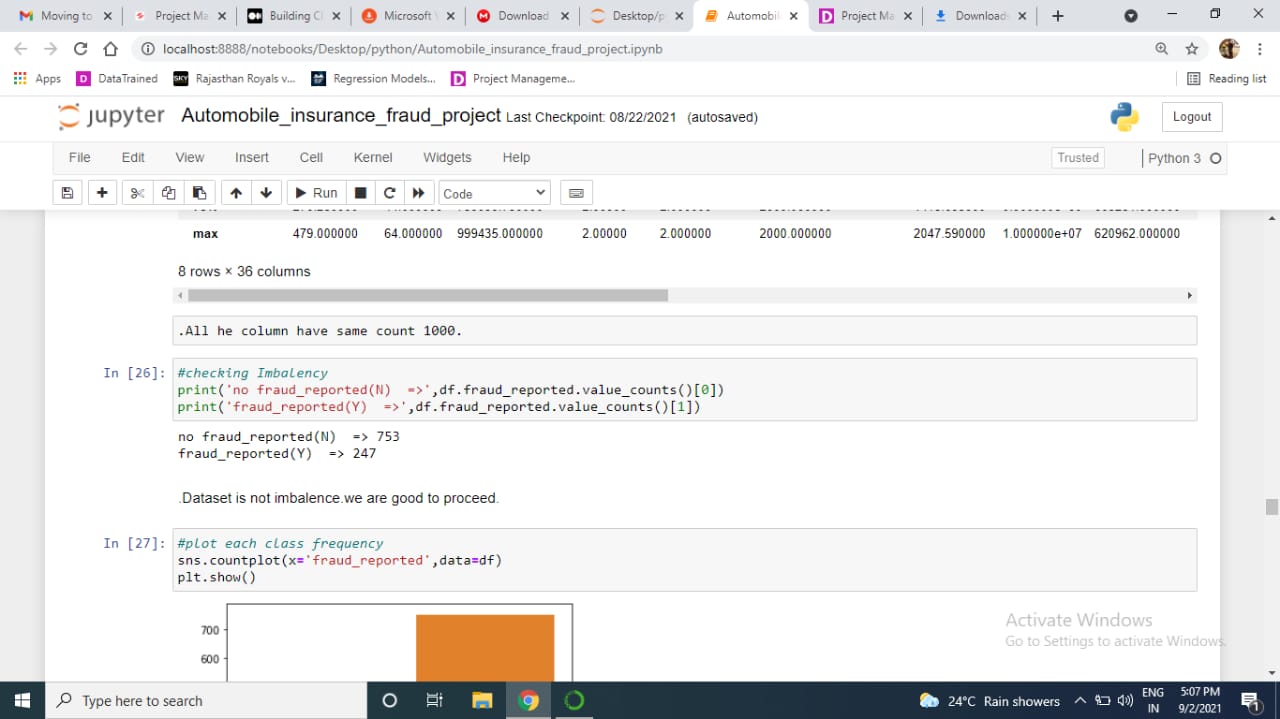
According to the dataset documentation, we need to remove the ‘\_C39’ as it contains only NaN values.





1. **Class Distribution**

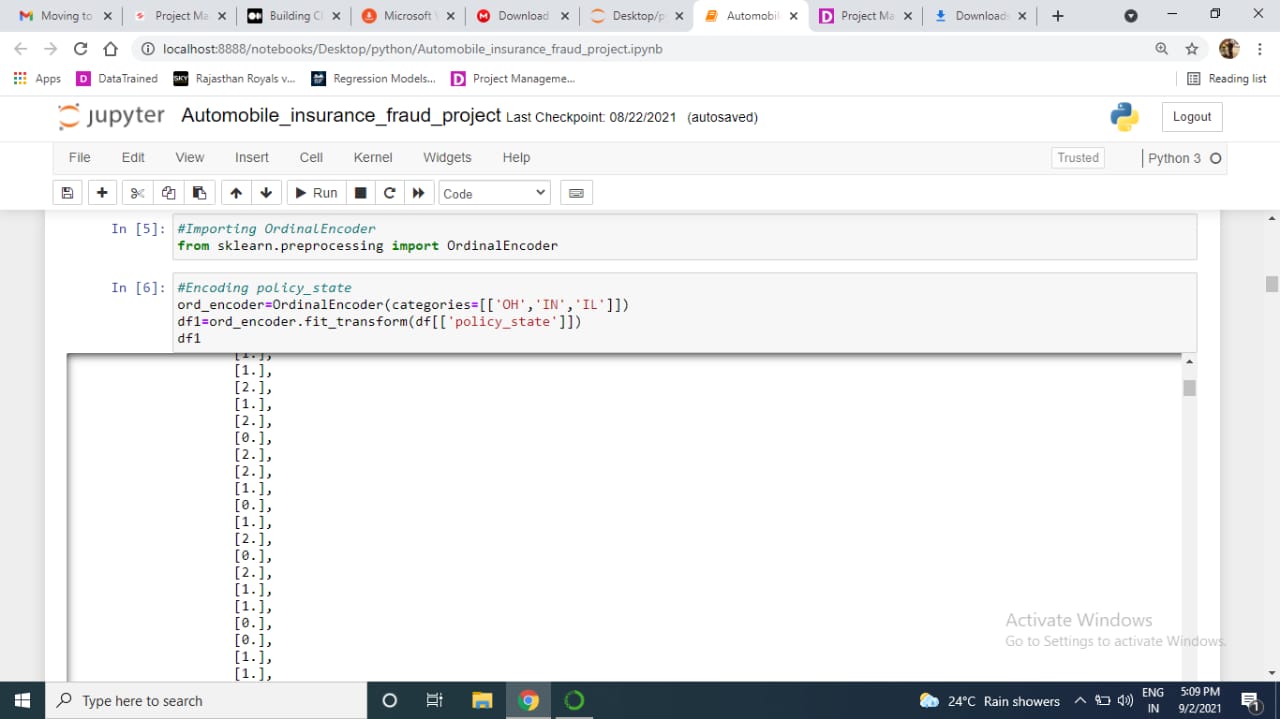
Another important thing is to make sure before feeding our data into the model is the class distribution of the data. In our case, where the expected class are divided into two outcomes, ‘Y’ and ‘N’, a class distribution of 60:40 can be considered ideal.

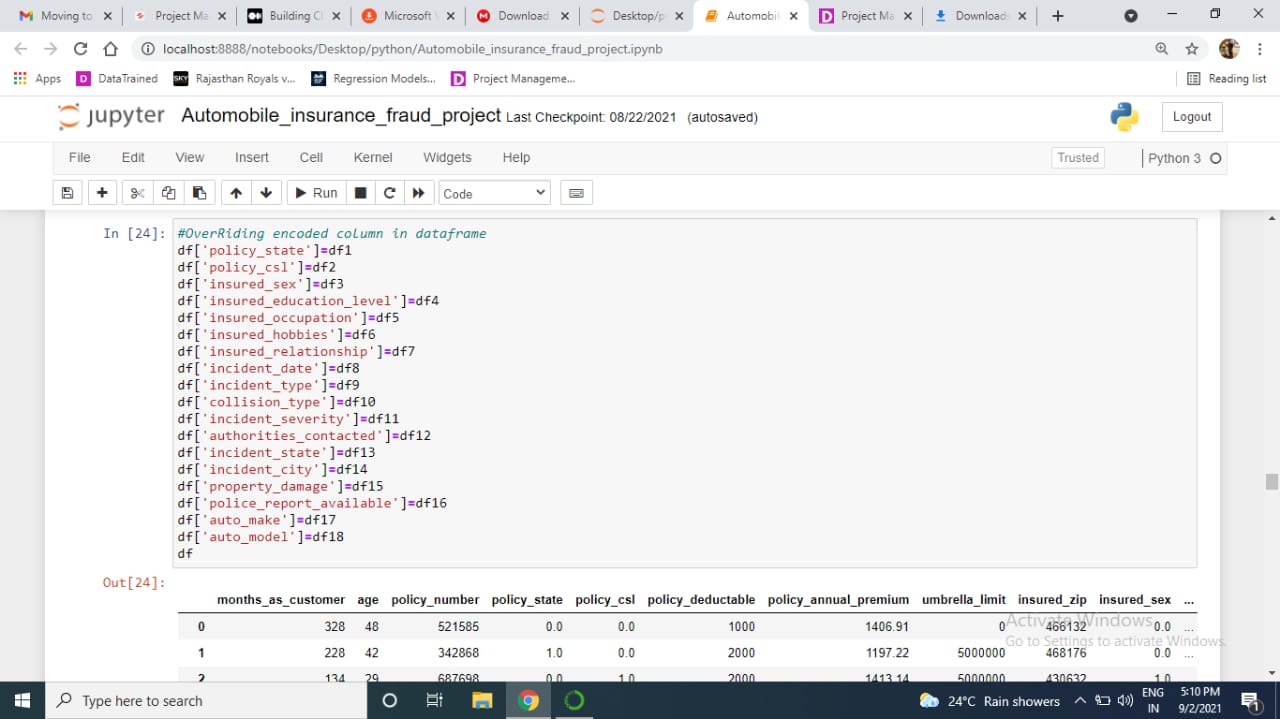


**Encoding Categorical Data**

Encoding all the column having object datatype into continuous datatype, as model only accept Numeric datatype.

After encoding using OrdinalEncoder provided by Sklearn, we can override the column to an existing dataset.



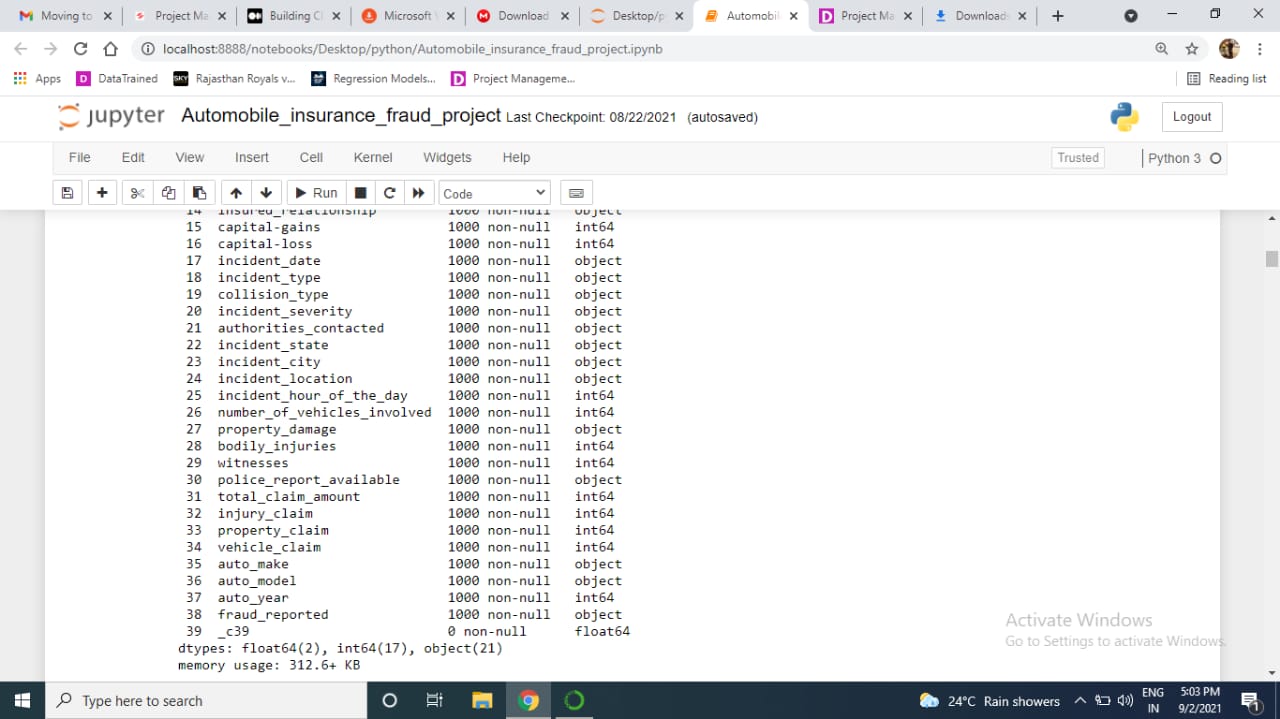


**Missing Values**

To check whether, the missing values are present in the column or not is also an important point before moving.

In some case, our data might have missing values in some column, this can be caused due to some reason such as human error.

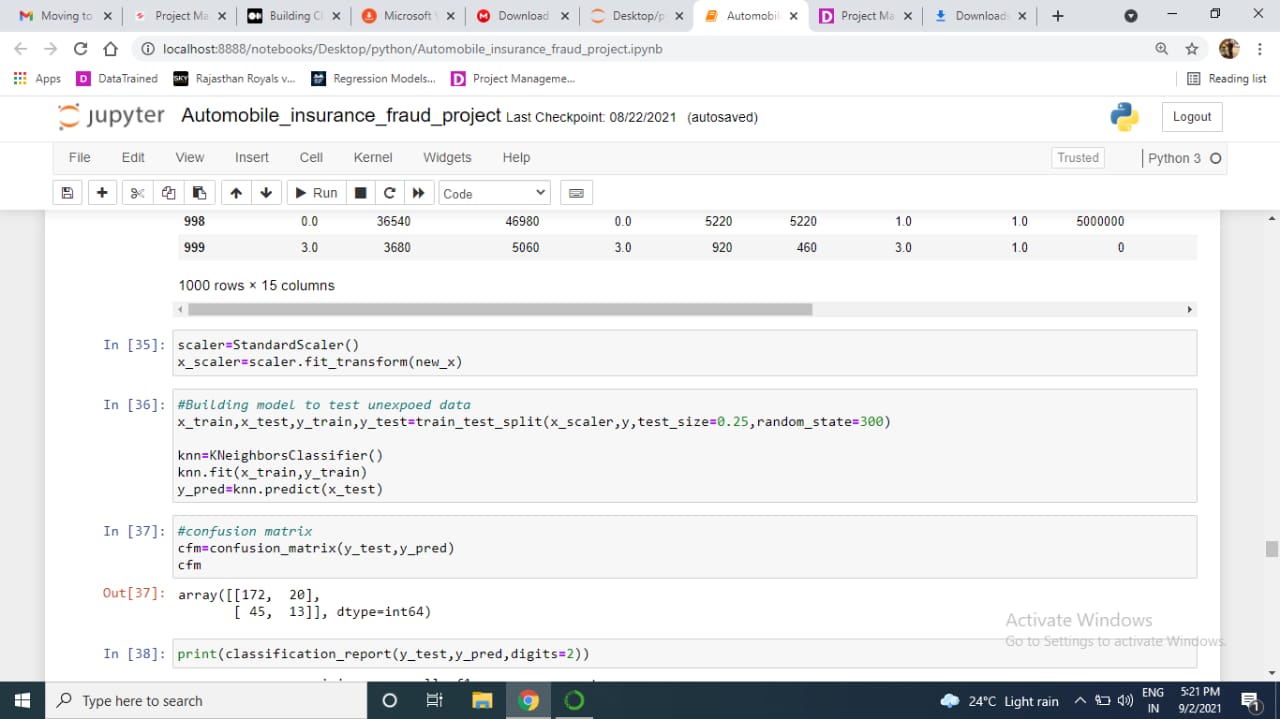
We can use the is-null( ) function from Pandas to check for any missing data and then use the sum( ) function to see the total of missing values in each column.



From the result, we can be assured that our data have no missing value and are good to go.

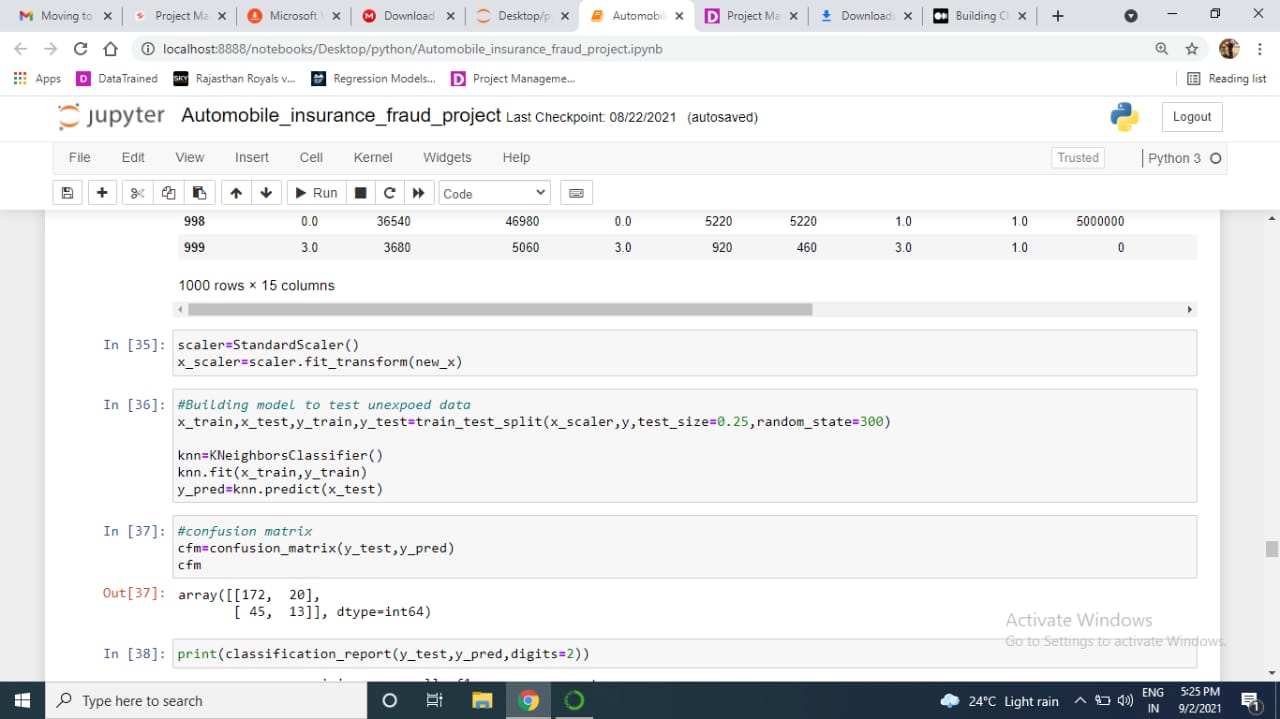
**Scaler Numeric Data**

After this, we will scale our numerical data to avoid outlier presence that can significantly affect our model. Using StandardScaler( ) function from Sklearn, we can scale our each columns that contains Numerical data.



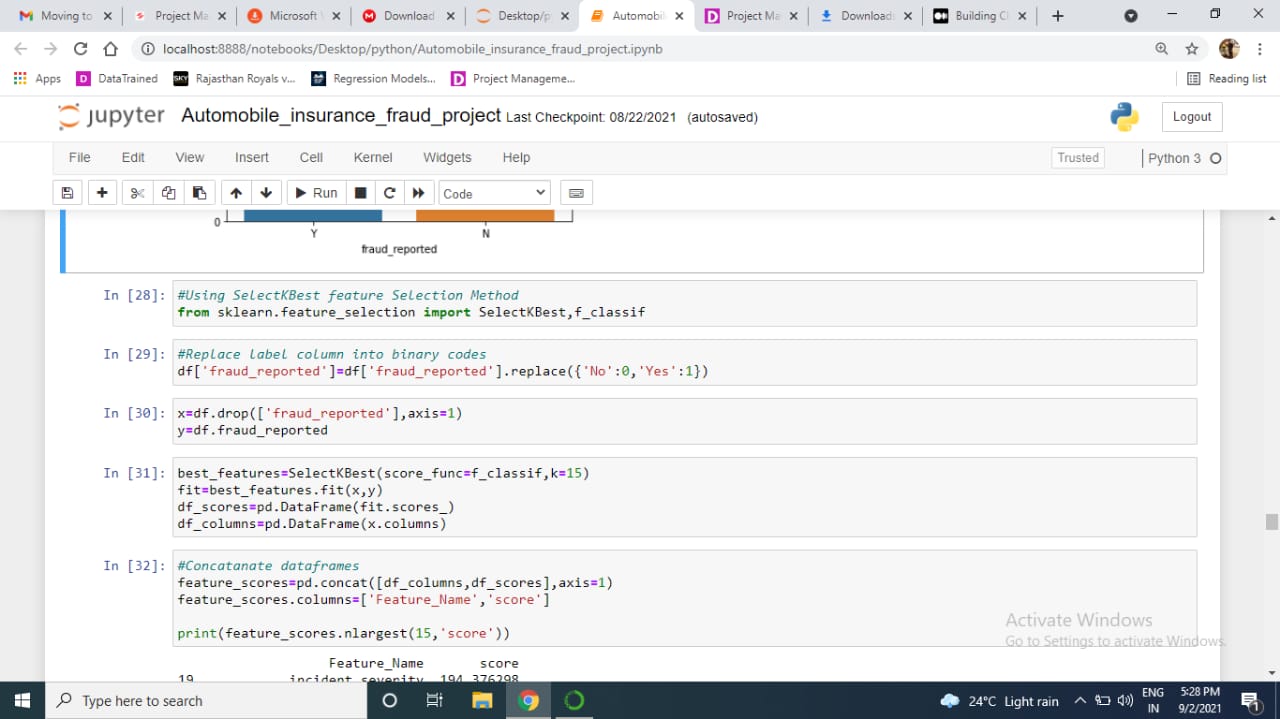
**Split Dataset for Training and Testing**

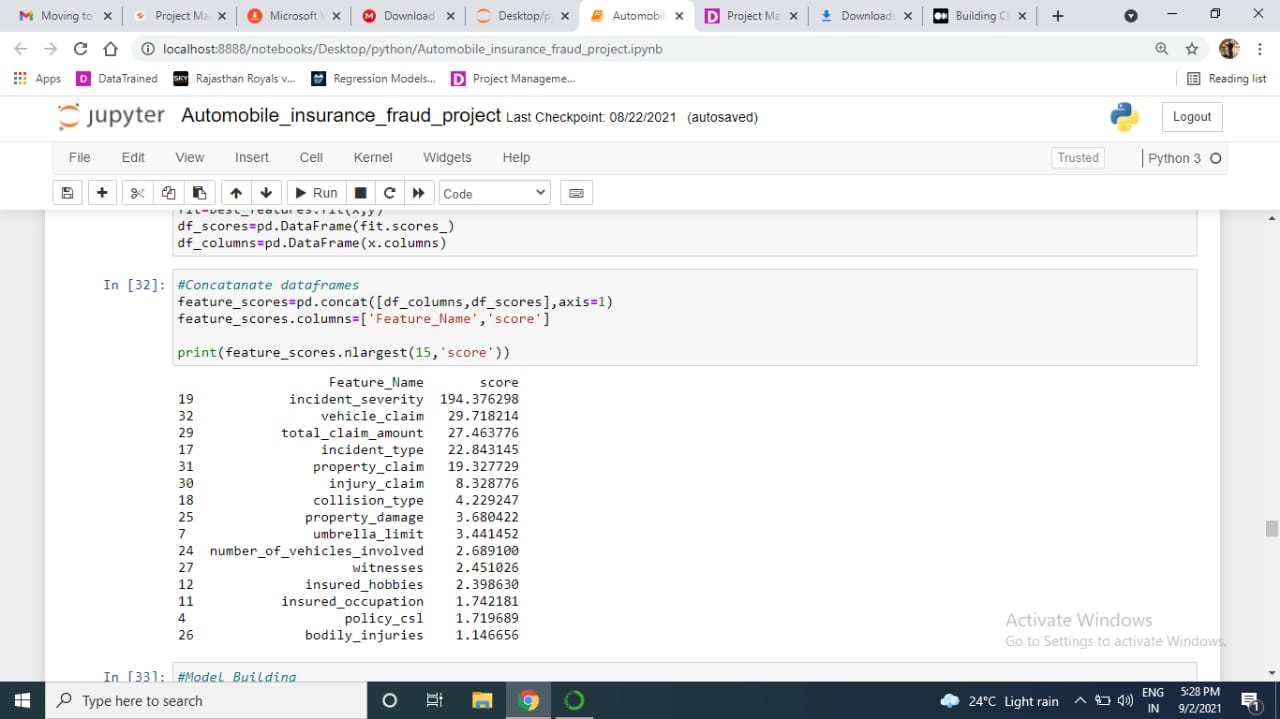
To finish up our data pre-processing steps, we will split our data into two dataset, X and Y. In this case, because we have enough data, we will split the data with ratio of 75:25 for training and testing respectively.



Selecting best features using SelectKBest and f\_class if

As we have very large no. of columns, so we can choose best column using SelectKBest function that will more affect the label. Those column that doesn’t affect more, we can reduce them.

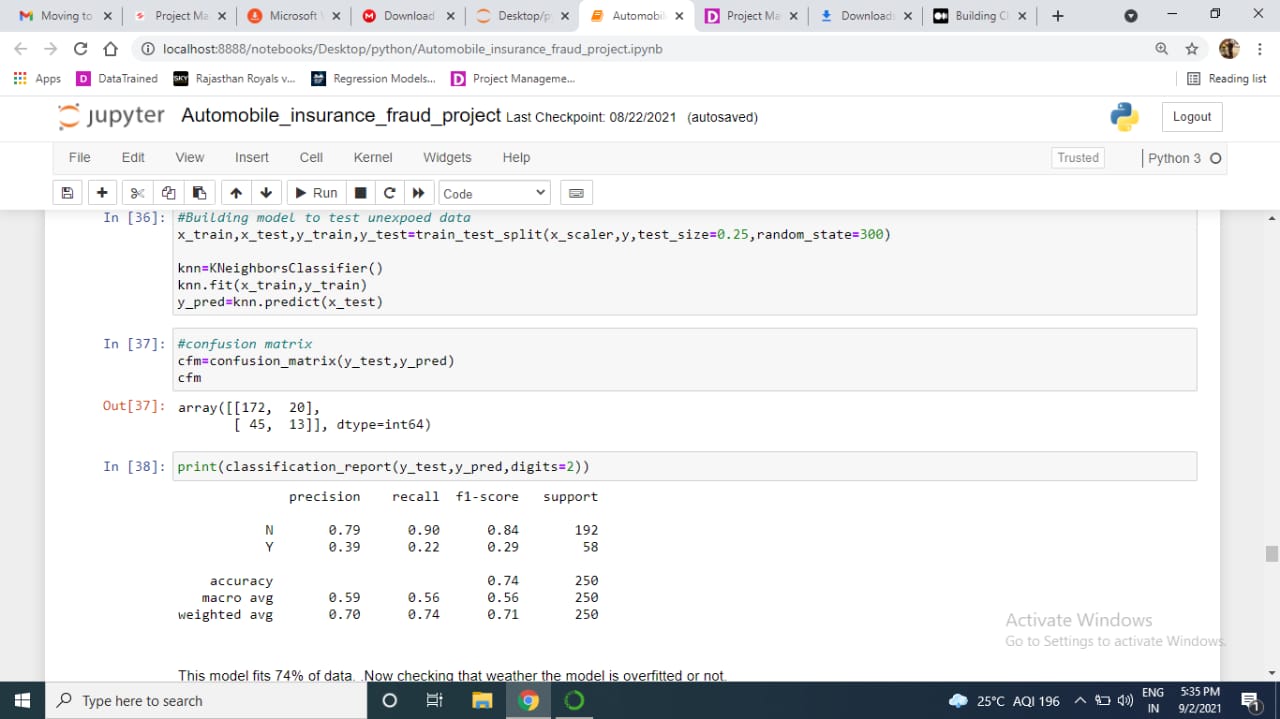




After this, we can make new df, that contains only those features.

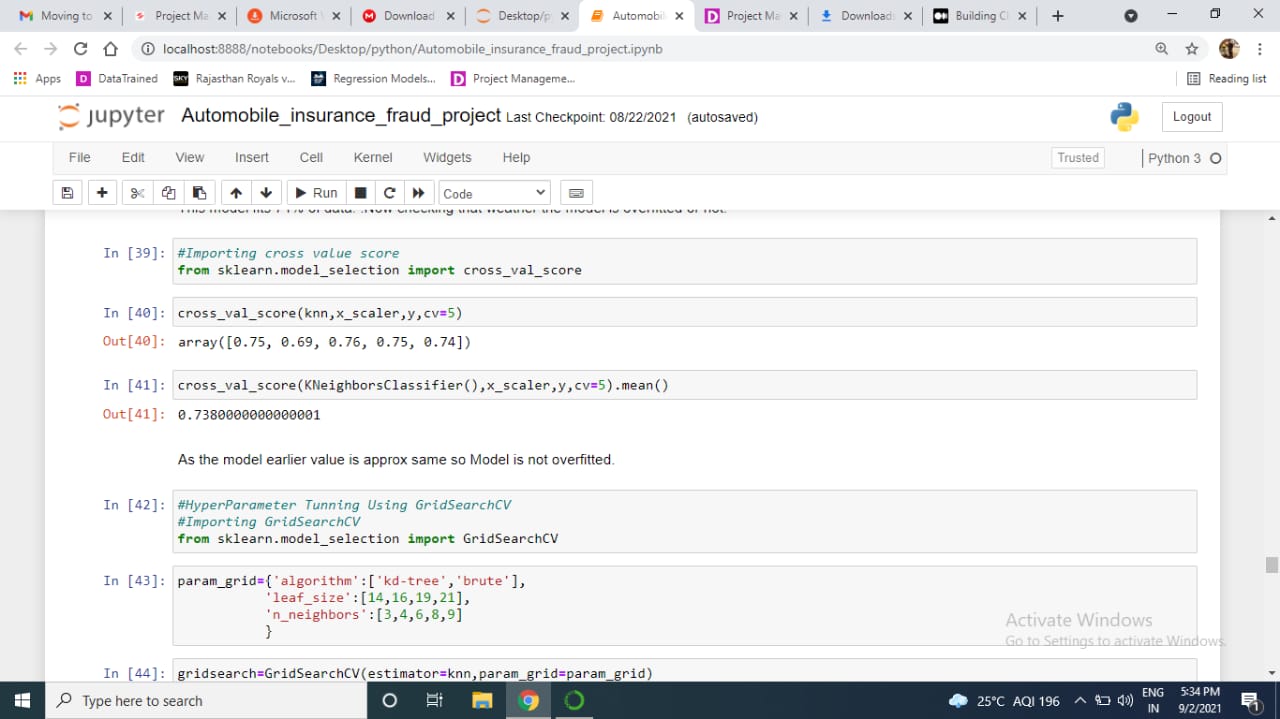
**Modeling**

After making sure our data is good and ready, we can continue building our model.In this case, we will first use KNeighborsClassifier algorithm to get the accuracy. We only get the accuracy of 75%, we got this with the help of confusion matrix and classification report.



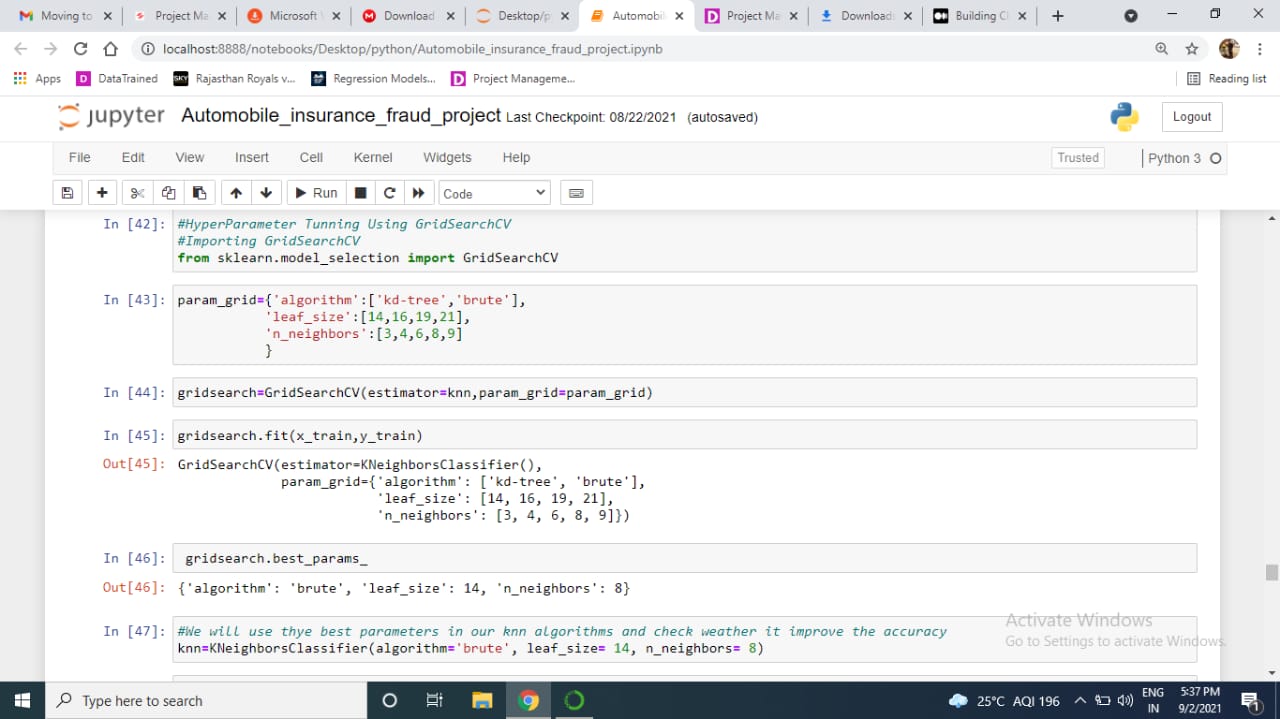
**Checking whether model is overtitled or not.**

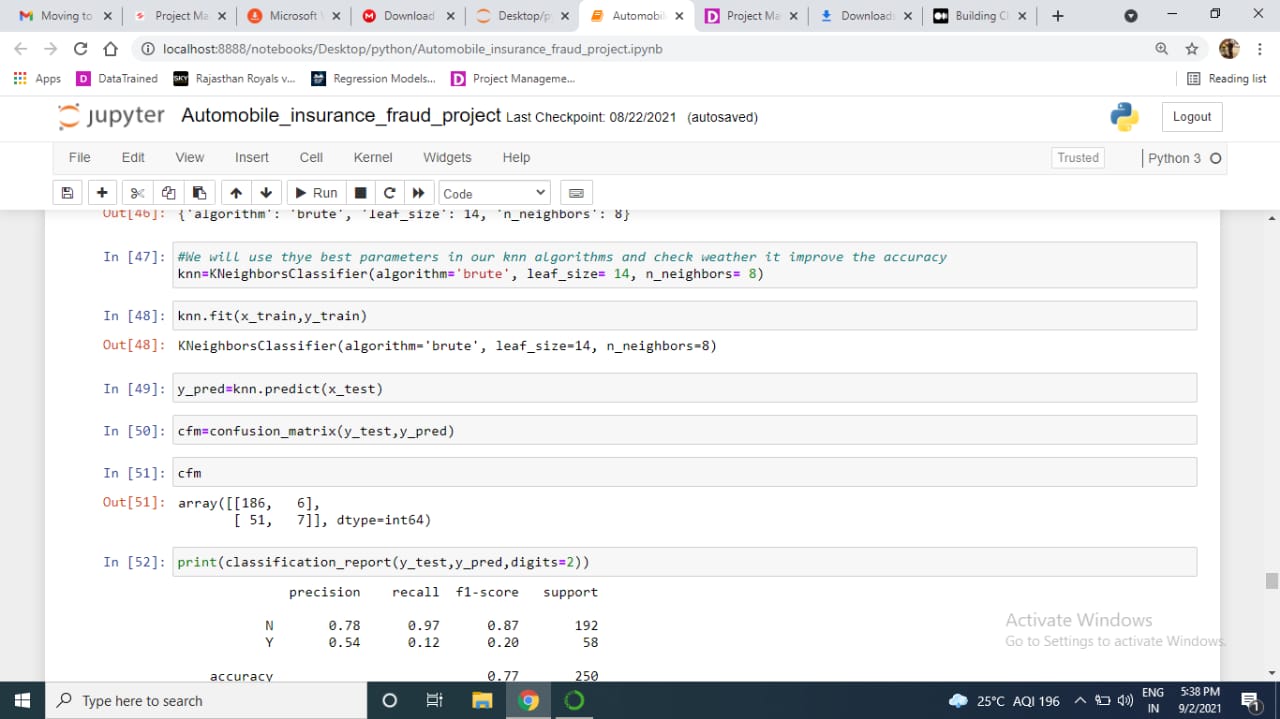
After training and predicting one model, we check that whether the model is overtitled or not using cross\_val\_score.



**HyperParameter Tunning**

After this, we will do hyperparameter tunning using’GridSearchCV’. After doing hyperParameter tunning, we again check the accuracy.

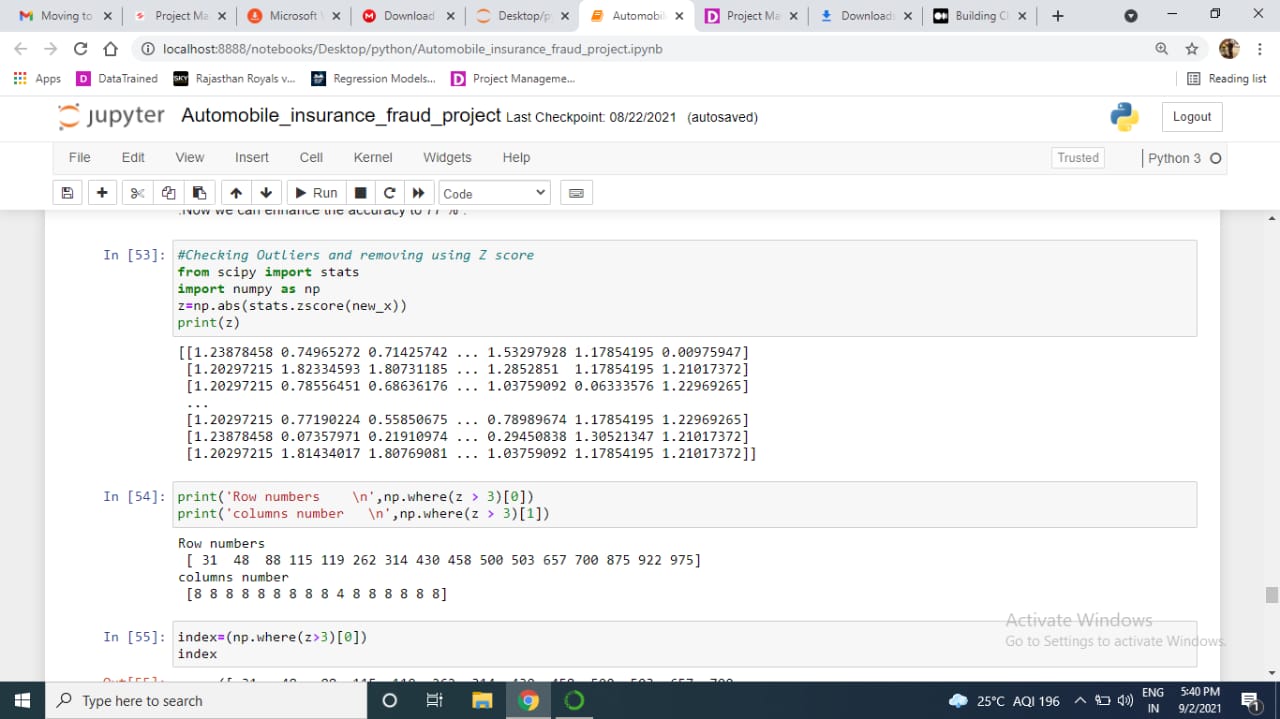




Accuracy enhances to 77%.

**Removing Outliers**

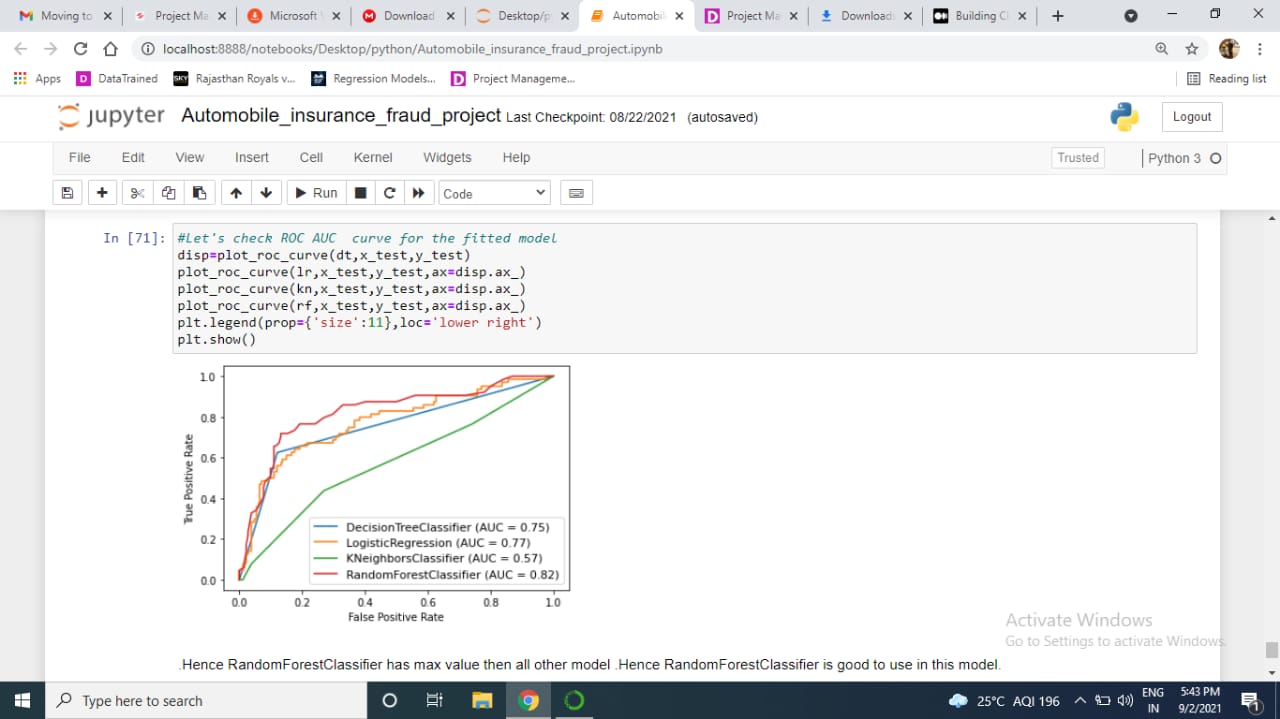
Now we check the accuracy after finding and removing outliers using Zscore.



After this, now the remaining rows are 984. We removed 16 rows that have outliers.

**Plotting ROC AUC to get best model**

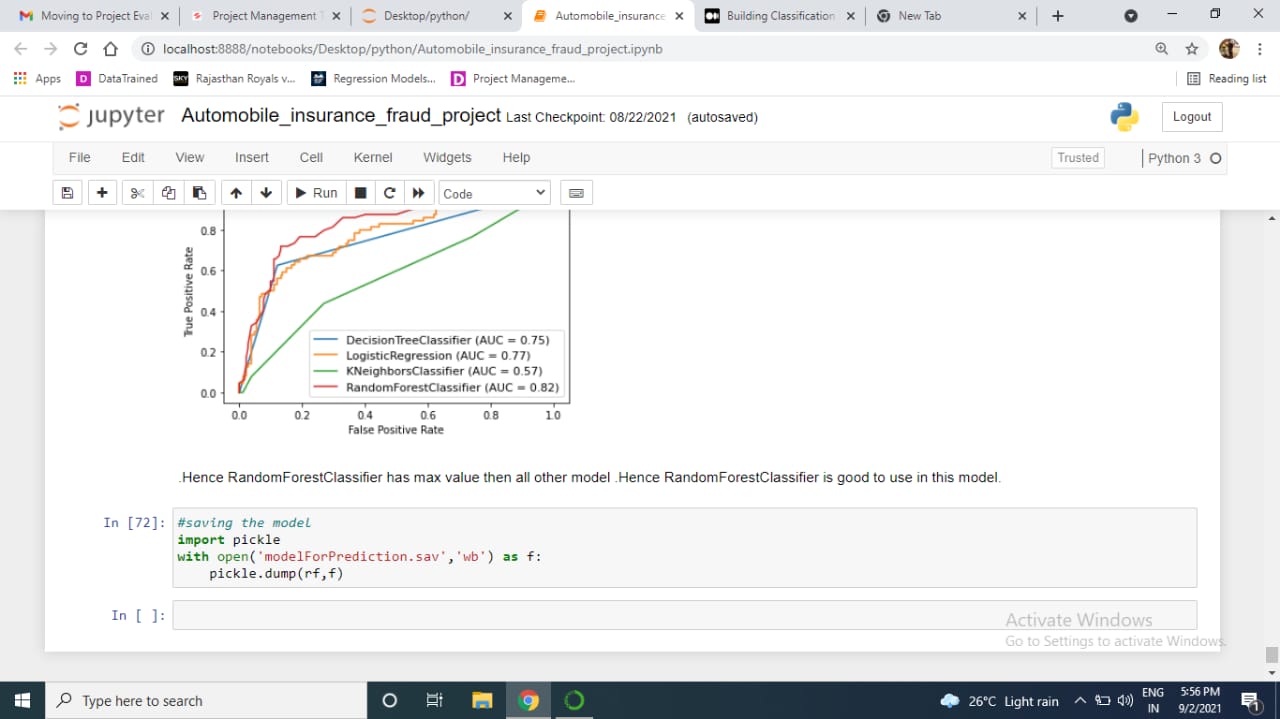
With the help of ROC AUC, we will get a comparison of the accuracy of different-different model using ROC AUC score.



From this screenshot, we saw that RandomForestClassifier has higher value i.e. 82%.

**Saving the Model**

At last, we can save model for further use.



**Conclusion**

For a simple model, we can see that our model did decently on classifying the data. But there are still some weakness as to do large no. of column, we have to drop some of them that cannot be encoded.

The result is not that much different after optimising the model using GridSearchCv, which can mean that we hit our limit with this model. To improve our performance we can try to look into another algorithm such as GradientBoostingClassifier.

Thankyou for reading, I hope you find it helpful!