**Insurance Claims – Fraud Detection**

**Problem Definition**

Fraud in insurance industry means make a fake claim for losing the property. Insurance fraud is a huge problem in the industry. It's difficult to identify fraud claims.

Machine Learning is in a unique position to help the Auto Insurance industry with this problem. The task is to create a predictive machine learning model that predicts if any insurance claim is that predicts if an insurance claim is fraudulent or not.

**Data Information**

In this dataset provided the details of insurance policy along with the customer details. It also has the details of the accident on the basis of which the claims have been made. The dataset is in csv format, its size is 261 KB. It consists of 1000 rows and 40 columns including target variable.

**Data Analysis**

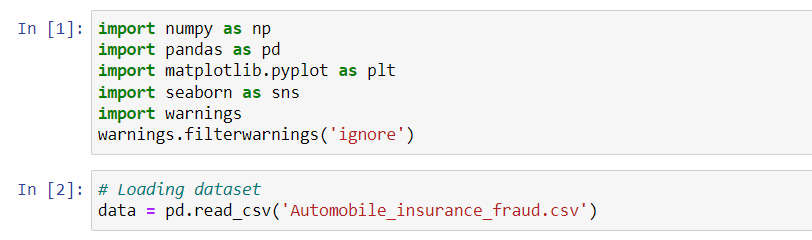
Data Collection

The data collected from an online data source (github.com). The data was received in the form of csv (Comma Separated Values) file.

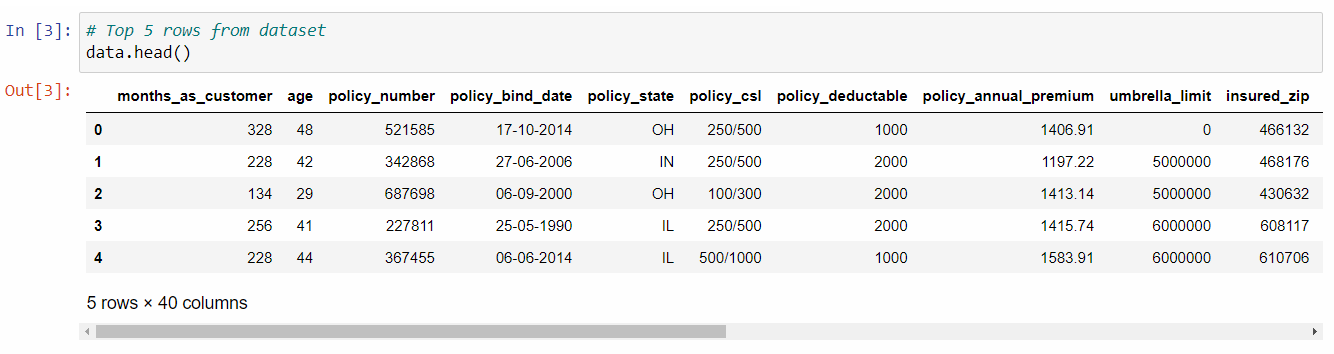
We have loaded the dataset in jupyter notebook with the help ofpandas, which is the most popular library that is used for data analysis and manipulation of data.

We have also imported numpy library which is mainly used for numerical calculations.

Matplotlib and Seaborn libraries are used to data visualization using following python commands:



We can see first 5 rows of dataset using data.head()



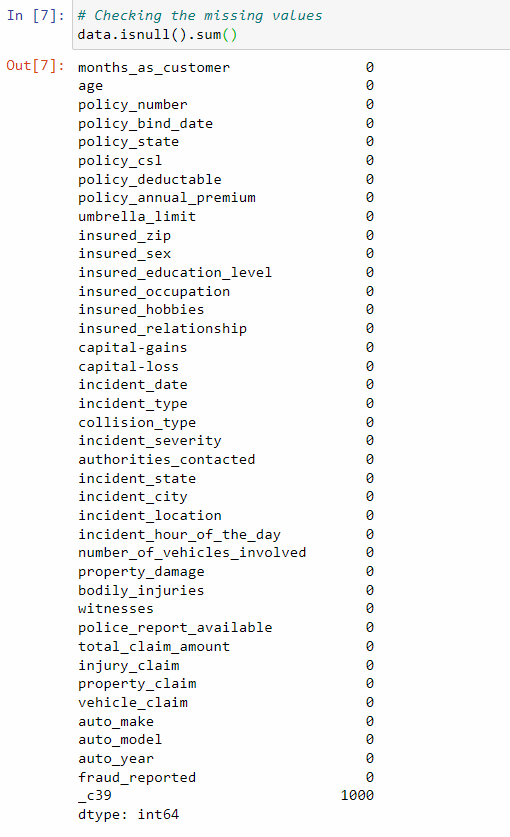
We can check the dimensions of the dataset using data.shape



We can see that the 1000 rows and 40 columns are present in the dataset. Out of 40 columns 39 columns are features and one column namely “fraud\_reported” is target variable (Label).

Data Cleaning and Data Preparation

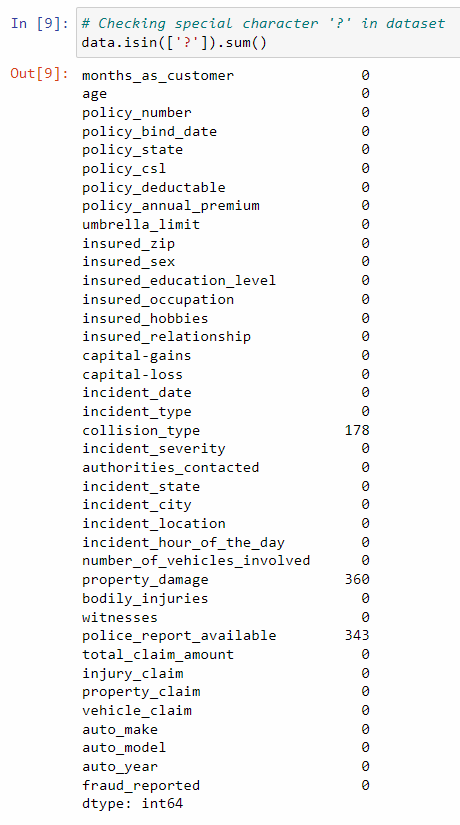
We can check the missing values in the dataset by using data.isnull().sum()



We can see that the column “\_c39” is completely empty, hence drop this column using command data.drop()



In dataset, we can observed in few columns special character “?” is available.

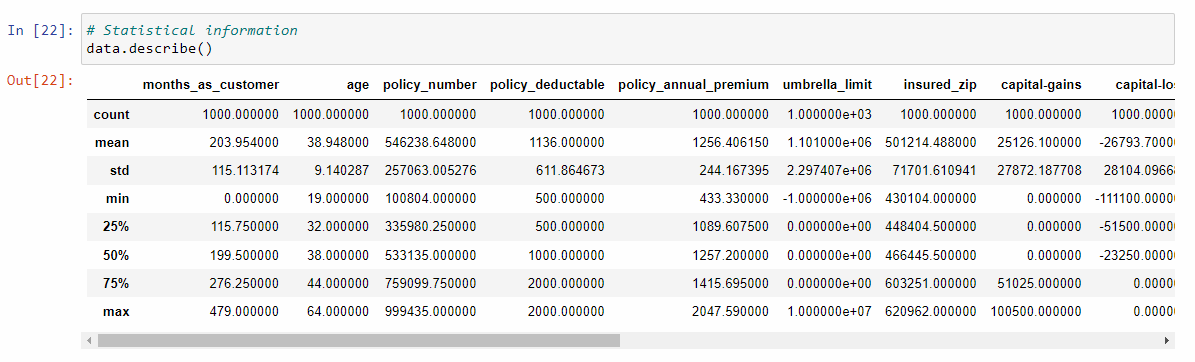


There are three columns namely 'collision\_type', 'property\_damage', and 'police\_report\_available' has consists of special character '?'.

We have filled with mode (Most frequent) value instead of special character “?”.

Summary of Statistics

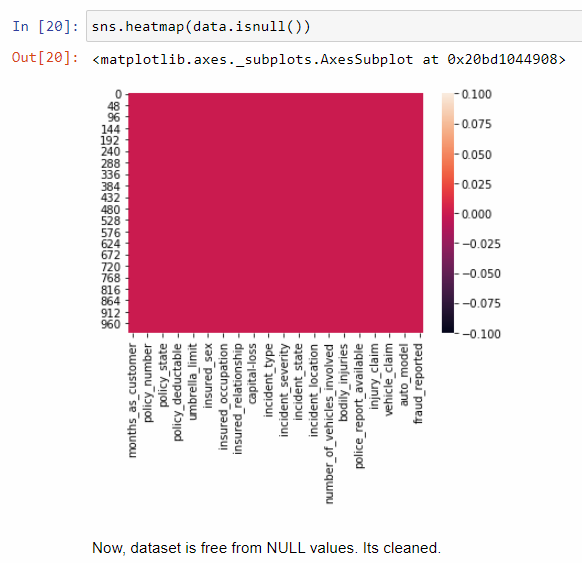
Now, using the following command data.describe() we can find count, mean, standard deviation, minimum, maximum, 25%, 50% (median), and 75% of the dataset which is used to decide that which column contribution more towards of model to take certain predictions and also used in testing of model.



* The difference between 75th percentile and max is not much, So there is presence of less Outliers.
* Mean and Median is almost same for every column, So the data is almost normal distributed.

Data Visualizations

After cleaning the data, there is no missing value available. We can this by using this command data.isnull().sum() and we can also see that from heat map.



Visualization can be divided into two categories.

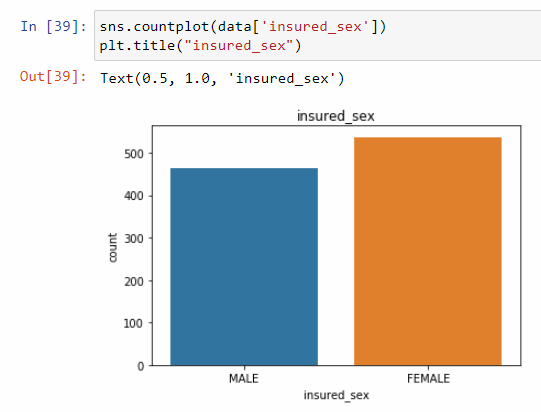
(1) Visualization of the Object (Sting) columns.

(2) Visualization of Numeric (int/float) columns.

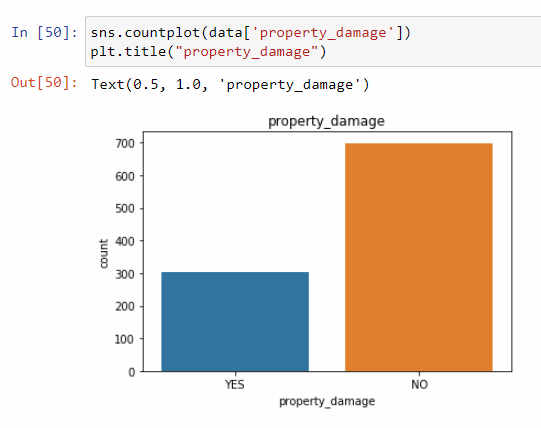
Visualization of Object (String) columns

For the string data, count plots are used as it gives the frequency of the columns.

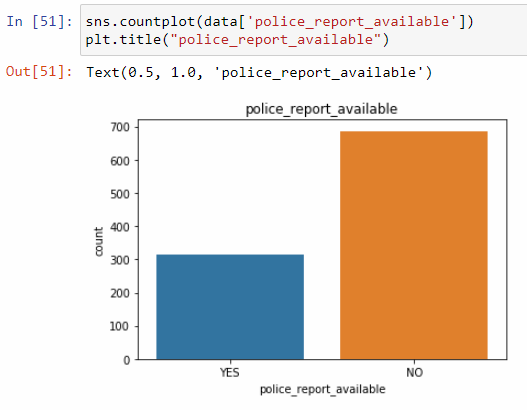
Observations made while analyzing data are:



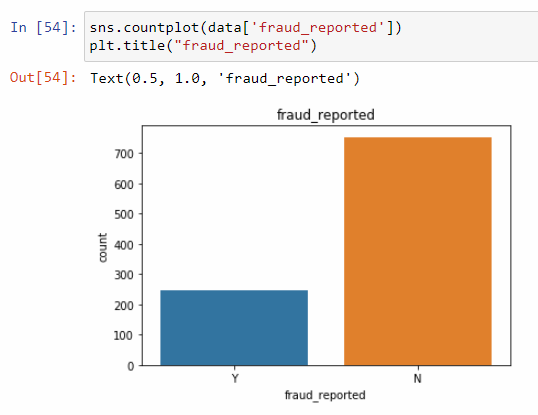
* Female members are more than male members.



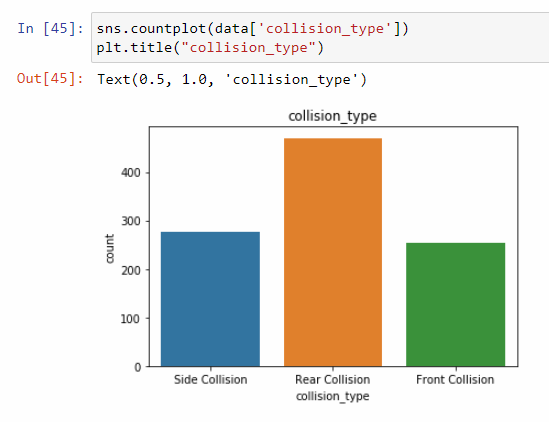
* 300 customer’s property has been damaged and around 700 customer’s property has not been damaged.



* Around 320 customer’s police reports are available and 680 customer’s police reports are not available.



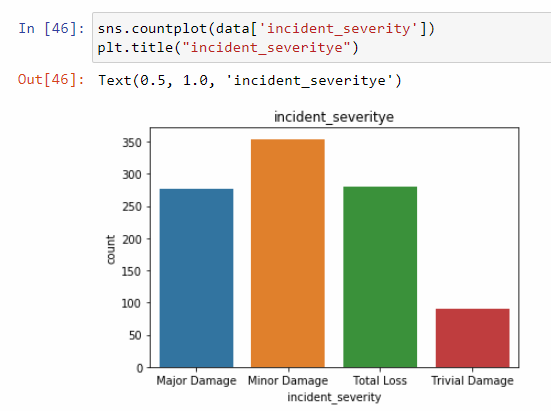
* More than 700 people have been reported with no fraud bur around 250 people have been reported with fraud.



* Rear Collision is the max collision type.



* Incidents are high in states like SC, NY and WV



* The incident severity is minor damage in most of the cases.
* There are too many categories in the column like , “auto\_model”, “auto\_make”, “incident\_location”. “incident\_date”, “insured\_occupation” and “insured\_hobbies”. Hence, it is difficult to conclude any observation out of these columns.

Visualization of Numeric (int/float) columns

We have used displot and scatter plot to understand data with int/float values. Observations made are:

* Variables “policy\_bind\_year”, “pl;ocy\_bind\_month”, “policy\_bind\_day”. “auto\_year” and “incident\_hour\_of\_the\_day” are showing values at a constant rate.
* Variable “umbrella\_limit” consists of maximum No. of Zero values.
* We could observe negative values in the “capital-loss” feature.
* “capital-gains” consists of values ranging till positive 100500.
* “capital-loss” consists of values ranging till negative 111100.

Label Encoding

Encoding all string values using LabelEncoder () to convert it numerical values.

EDA Concluding remarks

Performing complete EDA on the data (cleaning, integrating and transforming of data), we get a dataset with 1000 rows and 40 columns.

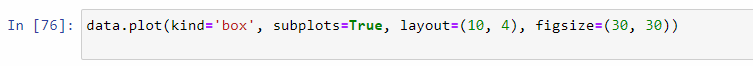
Concluding Observations are:

* The standard deviation of the variables in the dataset is very huge which means that the values in these columns are largely scattered and are not near means values. They are very far from their mean values.
* The values inside the dataset ranges from high negative values to high positive values. The value ranges are very high within the dataset.
* The min. and max values in every feature have huge range differences.
* Understanding data properly is difficult due to the huge no. of columns.
* The most negative correlated variable to the target variable is “incident \_severity”.
* The most positive correlated variable to the target variable is “vehicle\_claim”.
* High positive correlation was not observed in the data with respect to the target variable.
* There are 22 features which are positively correlated with the target variable.
* There are 17 features which are negatively correlated with the target variable.

Pre-Processing Pipeline

Outliers detections

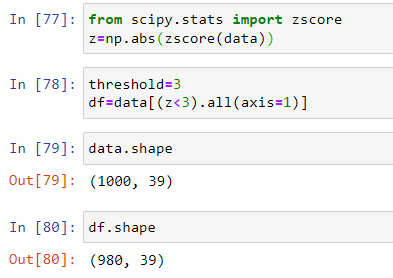
Now we will plot the outliers by using boxplot for all attributes as follows.



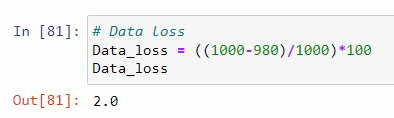
There are certain outliers are present in dataset.

Removing Outliers

In order to remove outliers from data, we have to use z-score method.

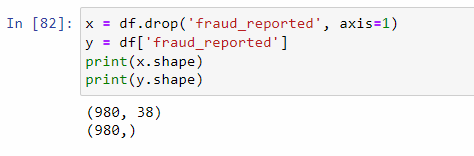


Now, we will calculate the data loss.



The data loss is 2% which is not much of huge data, Outliers are removed.

Splitting the dataset into x and y variables.



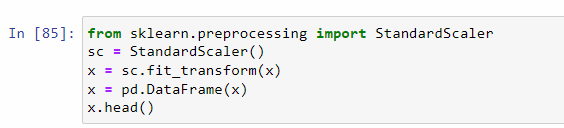
Skewness correction

Consider threshold value as +/- 0.5 as the range for skewness, we could se skewness by using python command df.skew()

The skewness for the required columns were resolved using power transform function.

Normalization

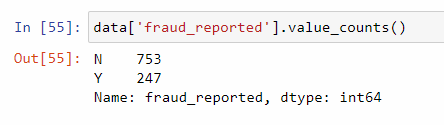
As the values in the dataset have high ranges, it becomes complex for a ML model to understand and read the data, hence data training becomes difficult which is not a proper way to deal with data to achieve good accuracy and get accurate predictions. Therefore, it is very important to normalize/standardize data which means getting data within certain range to have proper understanding of data. In this project we have used StandardScaler() techniques to normalize the data which brings data between the range of 0 to 1.



Sampling

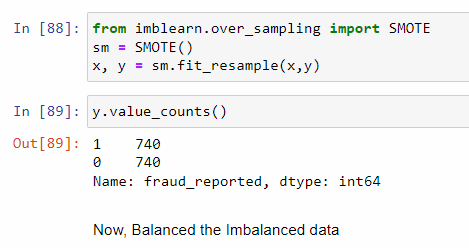
Imbalanced data refers to those types of datasets where the target variable has uneven distribution of the observations, i.e. one class label has very high No. of observations and the other has very low No. of observations.

In this dataset the target variable is imbalanced. Hence we can say this is imbalanced dataset.



Sampling is a technique to imbalance data to convert balanced data. There are several methods such as Choose proper Evaluation Metric, Resampling (Oversampling and Undersampling), Smote (Synthetic Minority Oversampling Technique) etc.

Here, we will use SMOTE technique to balance the imbalanced data.



Train Test Split

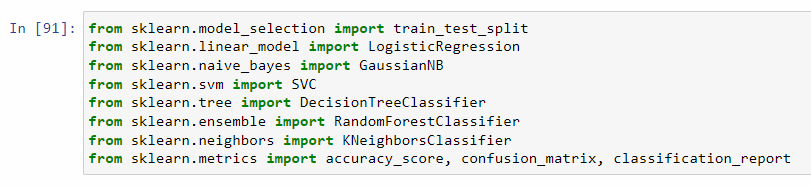
Using scikit learn library of python we have imported train test split to divide data into train data and test data. We have use test size is 0.2 (20%), rest of 0.8 (80%) data as train data. Random state provides us a seed to the random number generator by using following codes.

from sklearn.model\_selection import train\_train\_split

x\_train, x\_test, y\_train, y\_test = train\_train\_split (x, y, test\_size=0.2, random\_state=i)

Import Libraries & Metrics

Lets import all necessary libraries and metrics which is used to build the machine learning model and evaluation of models.



Already we have discussed about train test split. We will discuss all algorithms which has imported. Before that we will discuss briefly about accuracy score, confusion matrix and classification report.

Accuracy score

Accuracy is one of the matrix foe evaluating classification models. Informally, accuracy is the fraction of predictions our model got right.

Formally, accuracy has the following definition.

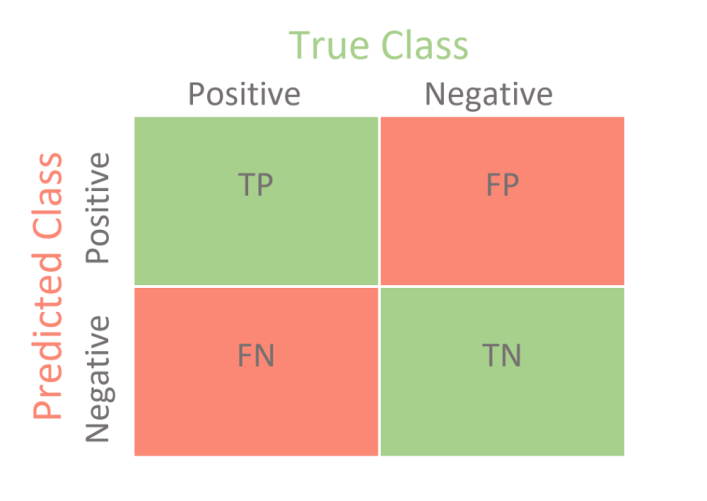
Accuracy score express in percentage.

Confusion matrix

Confusion matrix is a matrix used to determine the performance of classification models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itself can be easily understood but the related terminologies may be confusing, since it shows the errors in the model performance in the form of confusion matrix.

The matrix is divided into two dimensions that are Predicted values and Actual values along with the total number of predictions.

Predicted values are those values, which are predicted by the model, and Actual values are the true value for the given observations.



The above matrix is said to be confusion matrix, has the following cases.

* True Positive (TP): The model has predicted True (Yes or 1) and the actual value was also True (Yes or 1).
* False Positive (FP): The model has predicted True (Yes or 1) but the actual value was False (No or 0). It is called a Type-I error.
* False Negative (FN): The model has predicted False (No or 0) but the actual value was True (Yes or 1). It is called a Type-II error.
* True Negative (TN): The model has predicted False (No or 0) and the actual value was also False (No or 0).

Classification report

A classification report is a performance evaluation metric in machine learning. It is used to show model’s precision, recall. F1 Score and support.

Precision: Precision is defined as the ratio of true positive (TP) to the sum of true and false positives (TP+FP).

Recall: Recall is defined as the ratio of true positive (TP) to the sum of true positive (TP) and false negative (FN) . It is also known as Sensitivity.

F1 Score: The F1 is the harmonic mean of precision and recall. The closer the value of F1 Score is to 1.0 is the better expected performance of the model.

Support: Support is the number of actual occurrences of the class in the dataset. It just diagnoses the performance evaluation process.

Accuracy: The sum of true positives and true negatives (TP + TN) divided by the total number of samples. This is only accurate if the model is balanced. It will give inaccurate results if there is a class imbalance.

Macro average: A macro average will compute the metric independently for each class and then take average hence treating all classes equally.

Weighted average : The weighted average F1 score is calculated by take the mean of all per-class F1 scores while considering each class’s support.

Cross Validation

Cross validation is a technique for validating the model efficiency by training it on the subset of input data and testing on previously unseen subset of the input data. We can also say that it is a technique to check how statistical model generalizes to an independent dataset.

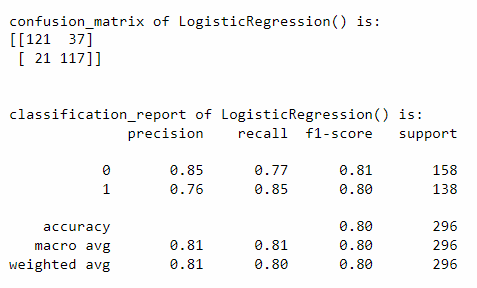
Logistic Regression

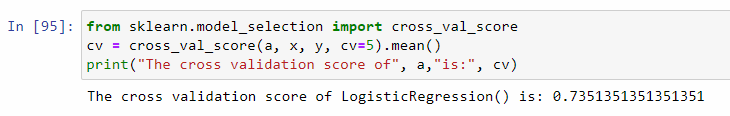
Logistic Regression is a supervised machine algorithm, which is used to solve classification problems.

The target variable of Logistic Regression is discrete (binary or ordinal). Predicted values Logistic Regression are the probability of the particular levels of the given values of the input variable.

Now, we will see how to perform Logistic Regression for this project.







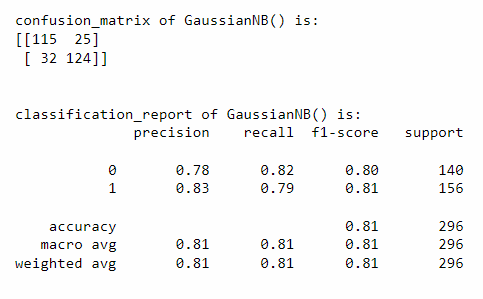
* In this project, the Logistic Regression accuracy is 80.40% and cross validation score is 73.51%

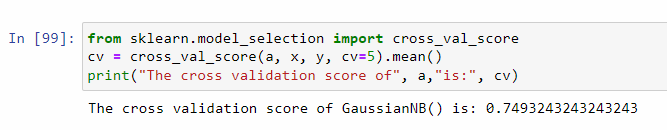
GaussianNB Classifier

Naive Bayes are the group of supervised machine classification algorithms based on the Bayes theorem. It is a simple classification technique, but has high functionality. They find use when the dimensionality of the inputs is high. Complex classification problems can also be implemented by using GaussianNB Classifier, generall known as Naïve Bayes Classifier.

Now, we will see how to perform GaussianNB Classifier for this project.







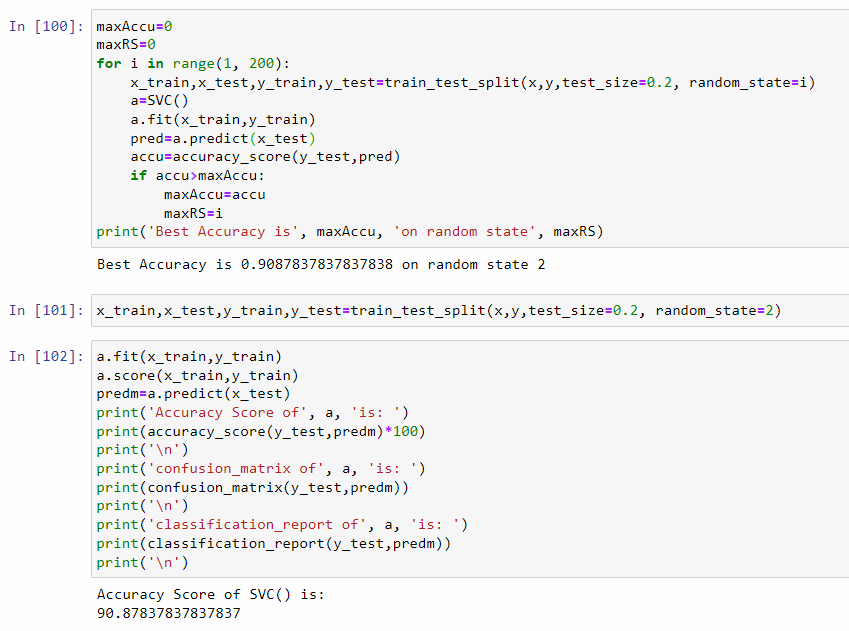
* The GaussianNB classifier accuracy is 80.74% and cross validation score is 74.93%

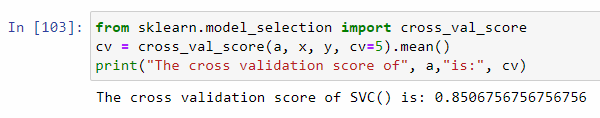
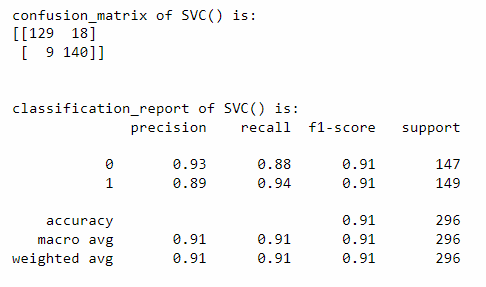
Support Vector Classifier

Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for both classification and regression problems. However, it is mostly used for classification problems. In the SVM algorithm, we plot each data item as point in n-dimensional space (where n is a No. of features you have) with the value of each feature being the value of particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.

Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier that best segregates the two classes (hyper-plane/line).

Now, check the performance of SVM classifier for this project.





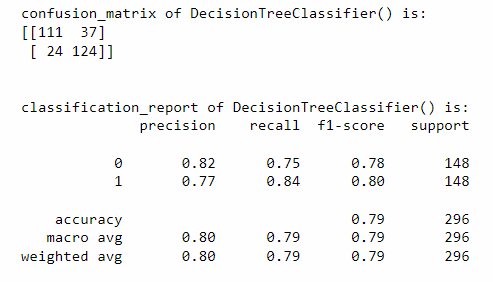
* The Support Vector Classifier accuracy is 90.87% and cross validation score is 85.06%

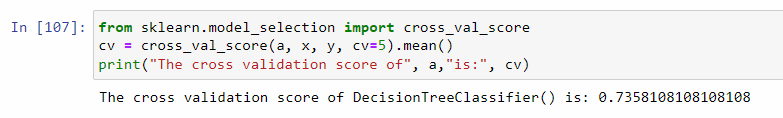
Decision Tree Classifier

Decision Tree is a supervised machine algorithm, which is used to solve for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

Let’s check the performance of Decision Tree Classifier with this problem.







* The Decision Tree Classifier accuracy is 79.39% and cross validation score is 73.58%

Random Forest Classifier

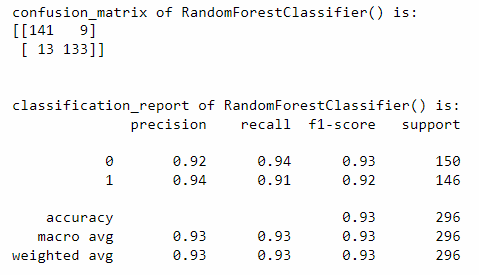
Random Forest Algorithm is a supervised machine learning algorithm, which is used to solve for both classification and Regression problems. It is based on the concept of ensemble learning, which is process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

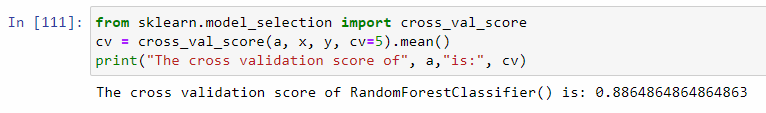
“Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset”. Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

NOTE: The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Now, check the performance of Random Forest classifier for this project.







* The Random Forest Classifier accuracy is 92.56% and cross validation score is 88.64%

KNeighbors Classifier

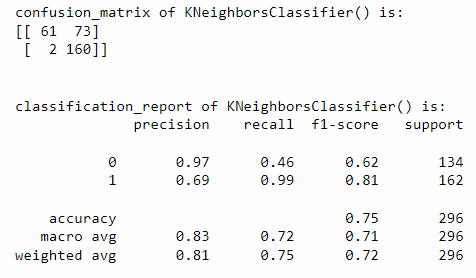
K-NN algorithm is a supervised machine learning algorithm, which is used to solve for both classification and Regression problems, but mostly it is preferred for solving Classification problems.

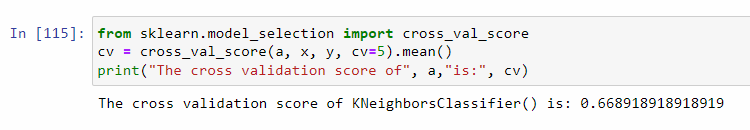
K-NN algorithm is a non-parametric algorithm, which means it does not make any assumption on underlying data.

It is also known as a lazy learner algorithm because it does not learn from the training set immediately instead it stores the data and at the time of classification, it performs an action on the dataset.

Let’s check the performance KNeighbors Classifier with this problem.







* The KNeighbors Classifier accuracy is 74.66% and cross validation score is 66.89%

NOTE: From the above algorithms, we can say that the Random Forest Classifier is working well by giving an accuracy of 92.56% and cross validation score of 88.64%. Now we will improve accuracy by Hyperparameter tuning by using GridSearchCV.

Hyperparameter Tuning

In order to increase the accuracy score of the model (Random Forest classifier), we use hyperparameter tuning of the best model in order to find best parameters by using GridSearchCV() in the following commands.



Conclude Remarks

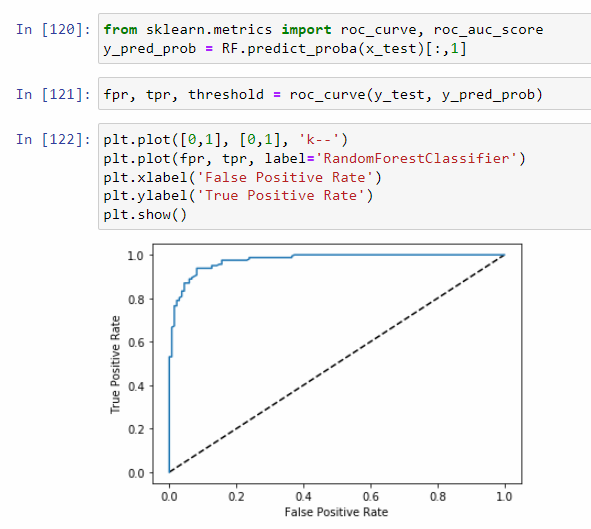
* After hyperparameter tuning the accuracy is decrease 92.56% and cross validation score is 88.64% to 91.55% and cross validation score is 89.18%.
* If we comparing all algorithm’s accuracy and cross validation score we have obtained the Random Forest Classifier accuracy and cross validation score is best. So, Random Forest Classifier is our final model for deployment.

AUC – ROC Curve

AUC – ROC (Area Under the Curve – Receiver Operating Characteristic) curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1.

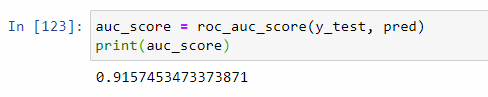
The ROC curve is plotted with TPR (True Positive Rate) against the FPR (False Positive Rate) where TPR is on the y-axis and FPR on the x-axis.

Now, plotting AUC – ROC curve by using python code.



As we see the graph is little far from 1 as our score is less than 1.

ROC AUC Score



* AUC Score is almost similar to the accuracy score as its 91.57%

Saving the model

In order to dump the model which we have developed so that we can use it to make predictions in future, we have saved or dumped the best model i.e. Random Forest Classifier using following lines of python code.

