Data description:

We have data in 3 different excel sheets. I have imported the data into the Jupyter notebook using the Pandas library. The 3 datasets and their respective columns are -

**Customers** – CustomerID, State, Salary, Gender, Smoker, Age, Occupation

**PolicyDetails** – PolicyID, CustomerID, Agent, Elimination Period, Income Replacement percent, Monthly Benefit, Annualized premium, product, policy effective date

**Claims** – PolicyID, Date of Loss, Diagnosis Category, Length of disability, Fraud, Amount paid, Amount recovered

|  |  |
| --- | --- |
| PolicyID | A unique ID identifying each policy |
| CustomerID | A unique ID identifying each insured |
| Gender | 1=Male, 0=Female |
| Smoker | 1=Smoker, 0=Non-smoker |
| Elimination Period | The number of days after the date of loss (i.e., the date of the accident, injury, or illness) that the policy goes into effect |
| Income Replacement Percent | The percent of your income the policy replaces |
| Monthly Benefit | The monthly amount the insured receives while on disability |
| Annualized Premium | The annualized amount of premium the insured pays to us |
| Latitude | The latitude of the insured's home address |
| Longitude | The longitude of the insured's home address |
| Product | The product the insured purchased (Disability or Life) |
| Policy Effective Date | The effective date of the policy |
| Date of Loss | This the date the accident, injury or illness occurred or was diagnosed |
| Diagnosis Category | The type of diagnosis |
| Length of Disability (in Days) | The number of days the insured collected disability |
| Fraud | An indicator of fraud. If "Yes", the claim was found to be fraudulent. If "No" the claim was not found to be fraudulent |
| Amount Paid | The amount of money the company paid the insured for the claim |
| Amount Recovered | The amount of money the company recovered on a fraudulent claim |
| Fraud | 1=Confirmed fraud, 0=No fraud confirmed |

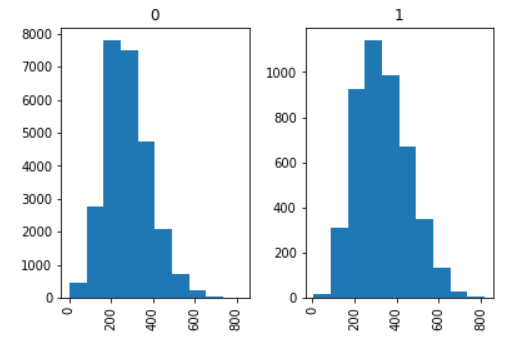
I inner joined customers and policydetails on ‘CustomerID’ and then, inner joined this data with Claims on ‘PolicyID’.

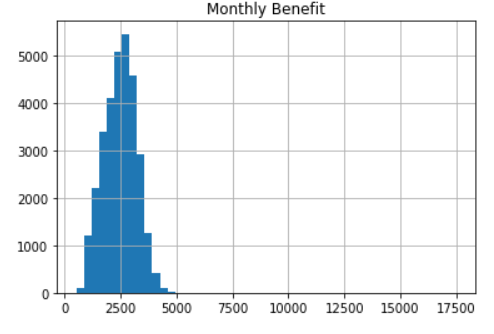
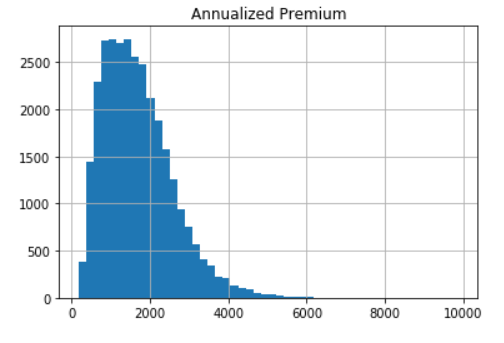
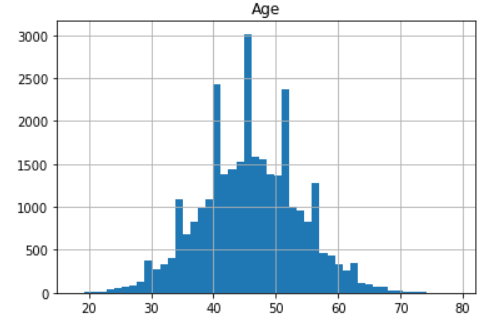
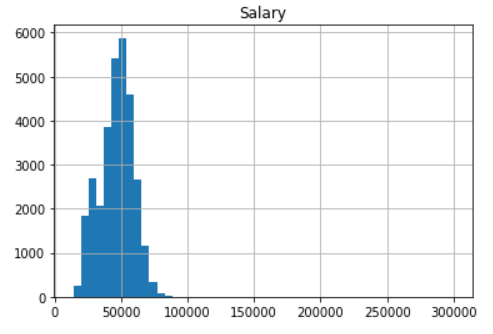
Data Preparation:

I filtered the policy data where we have cases only related to ‘Disability’ insurance. There are a few duplicated rows in the final data. I removed the duplicates and kept the first instance only. Later, I dropped irrelevant columns like PolicyID, CustomerID, date variables, etc. The final data has 30,838 rows and 12 columns. There are no null values. But there is a class imbalance.

|  |  |
| --- | --- |
| Fraud | 14.78% |
| Non-Fraud | 85.22% |

I want to see if fraudsters take advantage of the system and claim to have a longer disability period. But the distribution of the length of disability in days for fraudsters and non-fraudsters shows no significant difference. They have almost similar means and similar distribution.





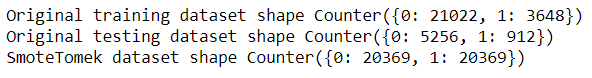
The distributions of the continuous variables are kind of normal in the data. But the scale of measurement is very different. This will cause an undue extra impact of some variables compared to the others while modeling some algorithms. So, it is advised to standardize the continuous variables. I used standardscaler in sklearn to standardize the variables.

Variables occupation, agent, elimination period and income replacement percent are categorical in nature. Using categorical variables as it is will cause problems in modeling. So, I created dummy variables using pandas. I dropped the original variables after the dummies were created. Now that the data is ready, I separated them into independent and target variables and split them to 80%:20% training and validation split. The purpose of creating a validation set is to ensure that our model is not overfitting.



Sampling:

To address the class imbalance issue, I used SMOTE oversampling followed by Tomek links cleaning. This will synthetically create new training instances of one level to match that of the other level. Cleaning is done by Tomek links. Now, the training data has 40,738 rows of equally distributed Fraud proportion.



Model: Naïve Bayes Classifier

**Naive Bayes** is an extremely fast algorithm relative to other [classification algorithms](https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2/?utm_source=blog&utm_medium=6stepsnaivebayesarticle). It is a [classification technique](https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2/?utm_source=blog&utm_medium=6stepsnaivebayesarticle) based on Bayes’ Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a feature in a class is unrelated to the presence of any other feature. In Gaussian Naive Bayes, each feature is assumed to be distributed normally.

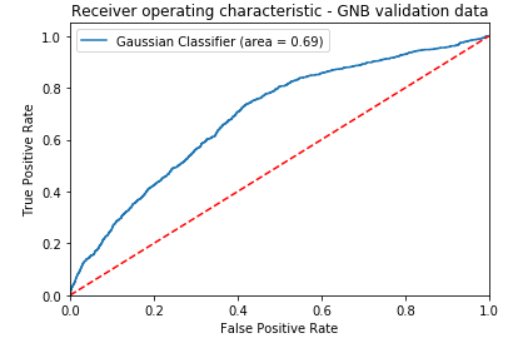
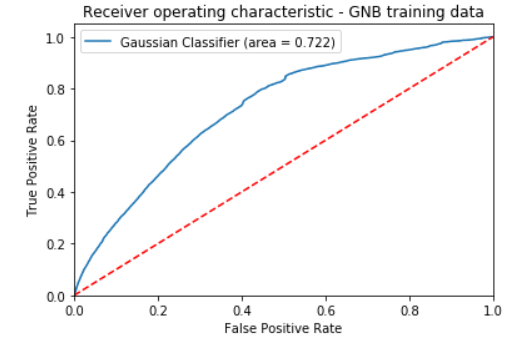
I ran Gaussian Naïve Bayes on the SMOTETomek over sampled training data and validated the model in the validation data. The results are published in the table below.

|  |  |  |
| --- | --- | --- |
| Metrics in % | Training data | Validation Data |
| Accuracy | 66.71 | 51.11 |
| Recall | 87 | 83 |
| Precision | 62 | 21 |
| AUC | 72 | 69 |
| F1 Score | 72 | 34 |

The accuracy of the training data is 66.71%, but it is low on validation data at 51.1%. This low accuracy on training data could be that the model is overfitting on the training data and is not that great at capturing the true trend. But the validation data is having a class imbalance. So, accuracy might not be the correct metric to look at. Sensitivity on the training and validation data is great. It is the most important metric to look at in this particular scenario. The validation precision has dropped compared to the training precision. This means our confidence in catching true positives has dropped and our model is flagging more other cases as true positives. The AUC of 69% says that the classifier does a decent, but not so great job of classifying fraud and non-fraud cases.

We can find alter the cut-offs to make the model highly sensitive in finding the true positives, but that is also increasing the false positives. Machine learning cant help beyond this point. It requires a business understanding for evaluation between what is the desired sensitivity of the model and what percent of false positives we can afford to achieve that sensitivity.

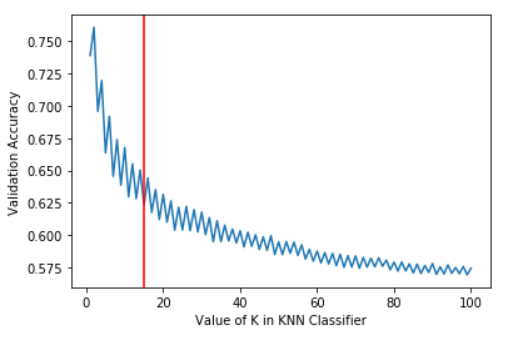
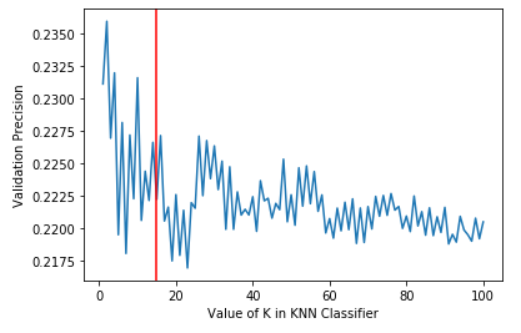
Overall, I find a need to bring more data to improve the accuracy of the model. More features that will account for Fraud cases need to be brought in the model.



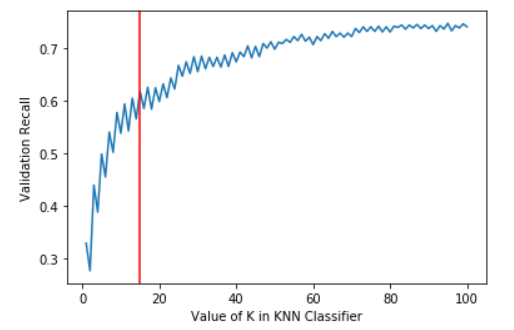
K Nearest Neighbors

KNN is a supervised parametric classification algorithm, just like GNB, but it doesn’t work on Bayes’ theorem. KNN models work by taking a data point and looking at the ‘k’ closest labeled data points. The data point is then assigned the label of the majority of the ‘k’ closest points. While this seems simple, several methods are available to compute the closeness. Euclidean distance is a common metric that is used for continuous variables. Since the data are seen how far each point is from another, large variables have undue impact. So, all the variable need to be standardized before running the model.

Choosing the optimal value of K is an optimization problem, as well as a part of bias-variance trade-off. If we choose a very low value of k, the classifier will have a very low bias but will be highly flexible. A high k value will yield results that have a high bias, but that are stable to data fluctuations. To subjectively decide k value, I have plotted the validation accuracy, recall and sensitivity of the model built over values of k from 1 to 100. Then, I decided a k value of 15 based on judgement.

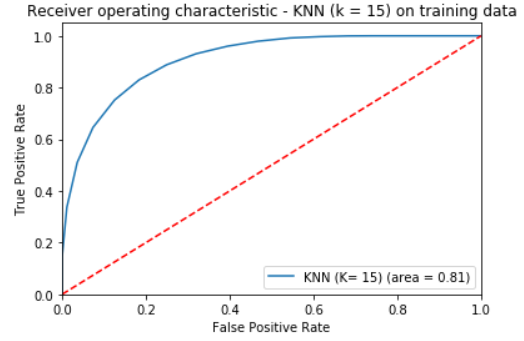
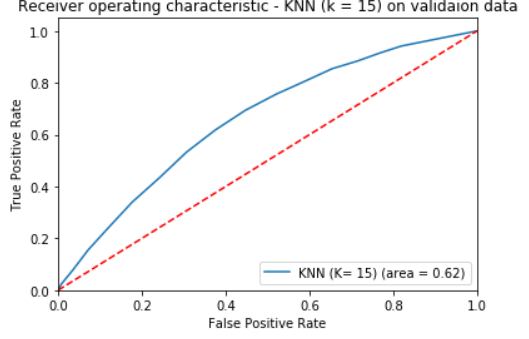
 

I choose k = 15 because I think it is where the recall curve has started becoming flat, whilst maintaining a good accuracy and precision.



Then, I built the final model with k =15 and calculated the metrics of importance for both training and validation data.

|  |  |  |
| --- | --- | --- |
| Metrics in % | Training data | Validation Data |
| Accuracy | 80.5 | 62.35 |
| Recall | 93 | 62 |
| Precision | 74 | 22 |
| AUC | 81 | 62 |
| F1 Score | 83 | 33 |



Looking the model performance statistics, we can say that the classifier probably is not doing any better than the simple GNB classifier, even though we are using a good k value. KNN (k=15) may be having a higher validation recall than GNB, but in terms of accuracy, precision and AUC, it is falling behind.

Support Vector Machine Classifier:

SVM is a powerful supervised machine learning model that utilizes the natural separation of the data in the feature space to identify target variable classes. The classifier separates data points using a hyperplane with the largest amount of margin. That's why an SVM classifier is also known as a discriminative classifier. SVM finds an optimal hyperplane which helps in classifying new data points. Support vectors are the data points, which are closest to the hyperplane. These points will define the separating line better by calculating margins. These points are more relevant to the construction of the classifier.  The kernel specified projects the non-linearly separable data in lower dimensions to linearly separable data in higher dimensions in such a way that data points belonging to different classes are allocated to different dimensions.

The have trained the SVM model with different kernel functions on the SMOTETomek oversampled training data and validated it on the validation data. Here are the results.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SVM (Sigmoid, C = 0.1) | | | SVM (Sigmoid, C = 0.01) | | |
| Metrics in % | Training data | Validation Data | Metrics in % | Training data | Validation Data |
| Accuracy | 55 | 55 | Accuracy | 62 | 60 |
| Recall | 55 | 56 | Recall | 63 | 59 |
| Precision | 55 | 18 | Precision | 61 | 20 |
| F1 Score | 55 | 27 | F1 Score | 62 | 30 |
| SVM (RBF, C = 0.1) | | | SVM (RBF, C = 0.3) | | |
| Metrics in % | Training data | Validation Data | Metrics in % | Training data | Validation Data |
| Accuracy | 69.3 | 60.5 | Accuracy | 70 | 61.3 |
| Recall | 79 | 75 | Recall | 80 | 73 |
| Precision | 66 | 24 | Precision | 67 | 24 |
| F1 Score | 72 | 36 | F1 Score | 73 | 36 |

RBF kernels have higher validation accuracy than sigmoid kernels for 2 different values of regularization parameter. Not only in accuracy, but also validation recall, precision and f1 score are higher for sigmoid kernels. This means that data are separable in higher dimensions in a radial direction than in a sigmoid way. We are managing to get higher validation recall but not precision. That means that the models are tagging more cases as positives, and only around a quarter of them are turning out to be true positives. It is a struggle between sensitivity and precision in all the models built on GNB, SVM and KNN. This means that there is a need to bring additional features that help us better predict the fraud.

Logistic Regression

Logistic Regression is a supervised parametric method that is used for classification purposes. I have built 3 models using 3 different regularization techniques in python, on the SMOTETomek oversampled data and validated the results on validation data. The results are as below. By using regularization, all 3 models are not overfitting. This can be seen from the almost same training and validation accuracy. The AUC on validation data is 0.73 which is the highest among all models built so far. The validation recall is also great. Precision followed the same pattern as previous models’ precision.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| L1 Regularizer | | | L2 Regularizer | | Elastic Net Regularizer | |
| Metrics in % | Training data | Validation Data | Training data | Validation Data | Training data | Validation Data |
| Accuracy | 68.5 | 64.5 | 68.5 | 64.6 | 68.5 | 64.6 |
| Recall | 73 | 73 | 73 | 73 | 73 | 73 |
| Precision | 67 | 26 | 67 | 26 | 67 | 26 |
| AUC | 75 | 73 | 75 | 73 | 76 | 73 |
| F1 Score | 70 | 38 | 70 | 38 | 70 | 38 |

