578d-Project Data Mining

Credit Card Fraud Detection

Brijesh Jethva
Faculty of Engineering
University of Victoria
Victoria, Canada

Deepak Kumar
Faculty of Engineering
University of Victoria
Victoria, Canada

Jaimin Modi
Faculty of Engineering
University of Victoria
Victoria, Canada

Manpreet Singh
Faculty of Engineering
University of Victoria
Victoria, Canada

***Abstract*—With the increase of credit card transactions, credit card frauds have been increasing. Fraud detection includes monitoring the behavior of the user in order to detect or avoid the undesirable behavior. The use of credit card is common in modern day society. Modern techniques based on data mining, machine learning, artificial intelligence and so n have been used to detect the credit card fraud. In our study, we have chosen logistic regression which is one of the data mining approach to train our data and then we have applied our results using support vector machine, random forests and decision trees and also created a confusion matrix based on true positive an false positive rate, to calcite the precision, recall and fallout. In this study precision and recall should be high and fallout should be low.**

Keywords—software engineering; data mining; machine learning; Fraud Detection;

I. PROJECT BACKGROUND

The use of credit card has seen enormous growth with time. With the rise in credit card use for both online and regular purchases, it's vulnerability to fraud is increasing dramatically. Fraud has become one of a major ethical issue related to credit card industry which has affected economy directly. Due to credit card fraud, billions of dollars are lost annually with nearly $4 billion estimated loss in 2014. In early days, stolen credit card was the most common fraud type but gradually with time due to increase in the popularity of e-commerce sites and with ease in online transactions, online credit card fraud is prevailing the most nowadays. Detecting such frauds is difficult task using the normal process, so the development of different credit card fraud detection has become of more importance.

Credit card frauds occur usually in two kinds, the first one is called Application fraud. In such kind, the fraudster tries to steal a new card from issuing company with the authorized person details or false information. The other kind of fraud is behavioral fraud. This kind of fraud is of four types namely card not found, mail theft, card holder not presents and counterfeit card. In Stolen/lost card fraud, the fraudsters steal or gets hold of someone's credit card and gain access to it. If we talk about mail theft fraud, in this the fraudster seize credit card in the mail before it reached to the authorized card holder. The most commonly occurring frauds include counterfeit card and 'cardholder not present' frauds. In such frauds, the details are accessed of the authorized person without his knowledge about the same. There are different ways in which information of the credit card holder can be accessed. These ways include unauthorized swipes, phishing, skimming. These frauds are conducted through mail, internet or phone.

Traditional methods of data analysis have also been used for the fraud detection. But these methods are complex and time consuming which deals with investigation of different domains like knowledge like financial, economics, business practices and law. Fraud often consists of many instances or incidents involving repeated transgressions using the same method. Fraud instances can be similar in content and appearance but usually are not identical. As the traditional methods were so time consuming and complex so machine learning and data mining techniques were introduced. In our study, we have taken the different classifier such as support vector machine, decision trees and we have train our data using logistic regression.

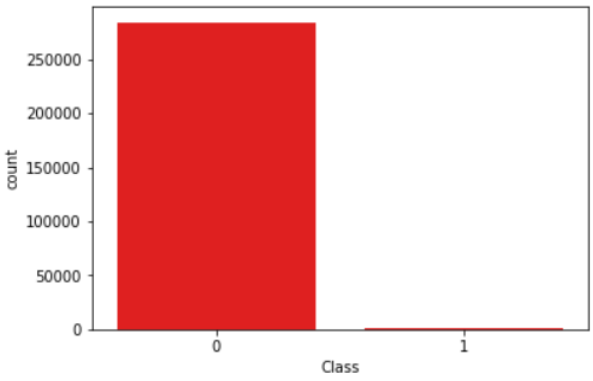
We choose this approach because this approach is one of the most commonly used practice in data mining. Logistic regression in our study is used for training the data. The other reason for choosing this approach is because our dataset is highly imbalanced so we have used logistic regression model as logistic regression model takes into consideration the high imbalance in data. This is used in predicting binary target variable which can either be zero or one, yes or no. Logic regression provides logistic curves.

II. DATA SOURCING AND DETAILS

In this study, we have taken the dataset from a website called Kaggle. The dataset was hosted by a user called Andrea.

The datasets contain transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where there are 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, the user cannot provide the original features and more background information about the data. Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependent cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.



Imbalance in Data

	Time	V1	V2	V3	V4	V5	V6	V7	\
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	

	V8	V9	...	V21	V22	V23	V24	\
0	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	
1	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	
2	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	

	V25	V26	V27	V28	Amount	Class
0	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0

Columns in Dataset

III. DATA MANIPULATION

The data has been trained on various models using numerous subsets of these features, but so far, no final feature set has been selected. It is clear, however, that some features appear to have a stronger correlation than others.

Data normalization is used to improve the data and combat the large variance between feature values. A simple feature rescaling formula (1) has been applied to certain attributes in order to constrain the data range between 0 and 1.

$$x' = \frac{x - min(x)}{max(x) - min(x)} \quad (1)$$

The group found that rescaling the features improved prediction accuracy between 5 to 10% on average when compared to non-normalized data on the same model. In dataset, the 'Amount' was not normalized, so before training the dataset and doing analysis, the amount has been normalized.

IV. MODELS EXPLORED

Since the classes are highly imbalanced, an appropriate measure of quality needs to be considered for fraud detection. Considering a simple model with all the classes (i.e. class 0 (non-fraud) and class 1 (fraud)), the model always predicted class 0 (non fraud). The precision achieved here is almost 99% because the ratio of class 0 to class 1 is highly imbalanced. But such a model would not be correct as it would not be able to classify with such type of model. It would not be able to predict whether the new class is fraud or not. The extremely high precision that it obtained by considering all classes as 0 is highly misleading. Hence, a more sophisticated way is needed to predict the future classes.

The approach used here is to try different sampling techniques to sample the data and then train each of the classifier with that sampling technique. A road map regarding what type of sampling techniques should be used and what kind of classifiers to be trained the data with is made. These are mentioned in the below graph. Each of the below sampling technique will be used on each of the 4 mentioned classifiers to see which of the classifier will have better Precision and Recall as compared to other techniques and classifiers.

Type of Sampling Techniques Used	Type of classifiers to be trained on Data
1. Under Sampling	1. Logistic Regression
2. Over Sampling	2. Support Vector Machines
3. Stratified K fold	3. Random Forest
4. SMOTE	4. Decision Tree

The definition for different accuracy matrices is as below:

Accuracy = TP+TN/Total

Precision = TP/ (TP+FP)

Recall = TP/ (TP+FN)

For the given dataset the TP, TN, FP and FN represented as below:

TP = True positive means no of fraud cases which are predicted fraud

TN = True negative means no of non-fraud cases which are predicted non-fraud

FP = False positive means no of non-fraud cases which are predicted fraud

FN= False Negative means no of fraud cases which are predicted non-fraud

Recall is given more focus for the current project because in this case the number of normal transactions will be very high than the number of fraud cases and sometime a fraud case will be predicted as normal. So, recall will give indication of only fraud cases.

UNDERSAMPLING

Since the data is imbalanced, one of the method used to balance the data is undersampling. When the number of samples n for minority class is made equal to the number the of samples n in majority class, it is called undersampling.

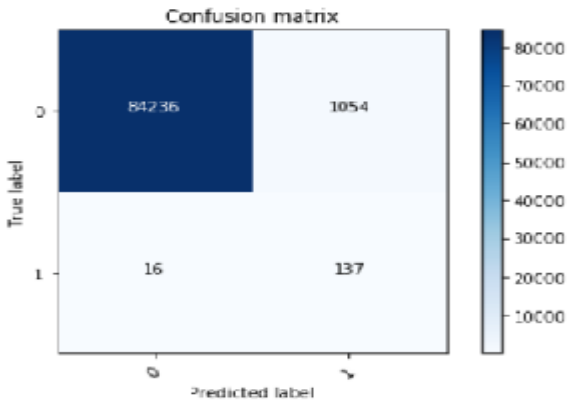
Here the ratios considered for undersampling are 1:1 (50:50), 2:1 (66.6:33.3), 3:1 (75:25). The models considered for evaluation are Logistics Regression, SVM and Random Forest. The best results in terms of least number of False

positive were obtained for the 3:1 ratio. The confusion matrix for all three models are as given in screenshots below.

Undersampling can solve the class imbalance issue and increase the sensitivity of the model. However, we also risk removing some of the majority class instances which is more representative, thus discarding the useful information.

Logistic Regression:

Logistic Regression is a type of classification algorithm involving a linear discriminant. Unlike actual regression, logistic regression does not try to predict the value of a numeric variable given a set of inputs. Instead, the output is a probability that the given input point belongs to a certain class.

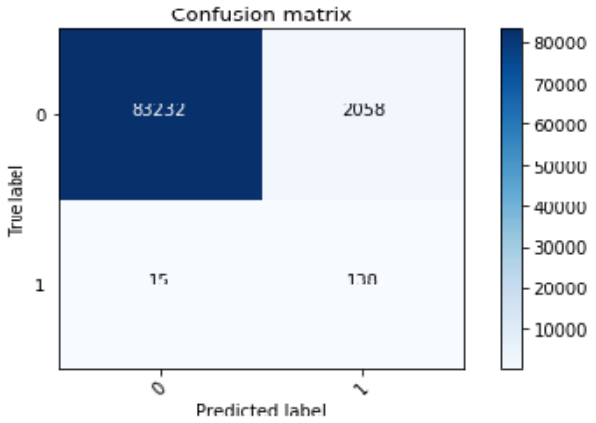



```
Precision =     0.115
Recall (TPR) =  0.895
Fallout (FPR) = 1.236e-02
Average Precision 7.89323782335 %
Average Recall 92.1021282076 %
Average Fallout 2.61555267796 %
```

Confusion Matrix for Logistic Regression

Support Vector Machines:

Support vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

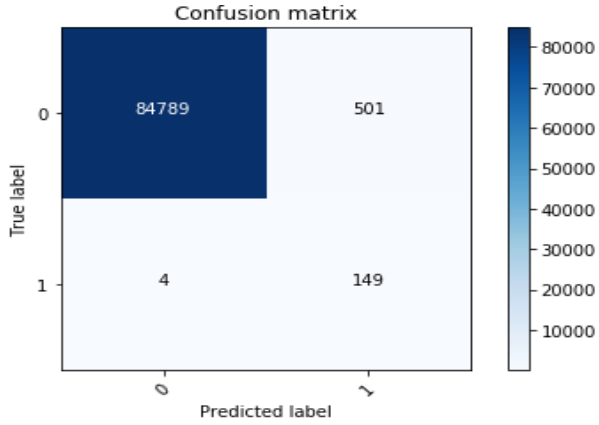



```
Precision =     0.063
Recall (TPR) =  0.902
Fallout (FPR) = 2.413e-02
Average Precision 7.69210222111 %
Average Recall 91.8638719856 %
Average Fallout 2.59022660236 %
```

Confusion Matrix for SVM

Random Forest Classifier:

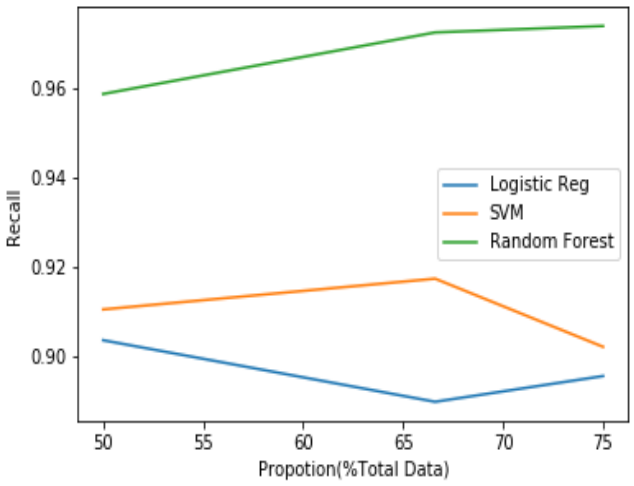
A random forest is a Meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.




```
Precision =     0.229
Recall (TPR) =  0.974
Fallout (FPR) = 5.874e-03
Average Precision 9.38443274355 %
Average Recall 92.4773996444 %
Average Fallout 2.3676911652 %
```

Confusion Matrix for Random Forest

The comparison for recall value for all three model is as below.



Comparison for 3 models in undersampling

As seen from the plots, the best values are obtained at proportion 75 percent. Also, from the graph it can be seen that from the 3 classifier Random Forest classifier gives the best result. However, the precision is very less which means the model is predicting other class wrong. That is the model is catching fraud transactions but it is also catching innocent transaction which are not fraud.

OVERSAMPLING

OVERSAMPLING duplicates the samples of minority class such that its proportion becomes almost equal to that of majority class samples. This way the data becomes balanced. The downside of oversampling is that because of duplicate data the model overfits the sample. It also because difficult to perform a cross validation because of oversampling.

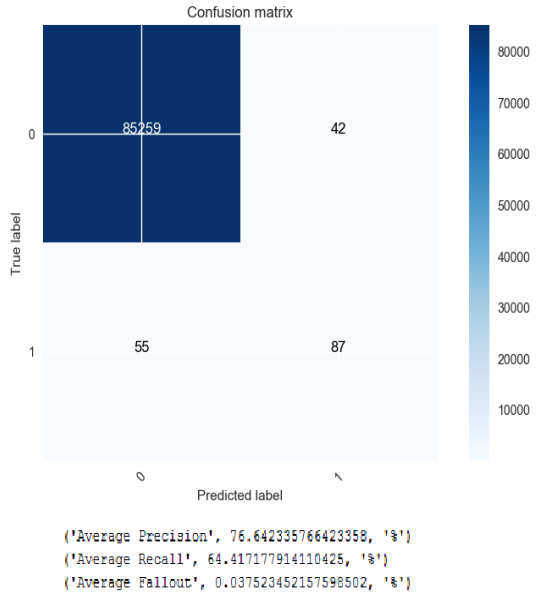
The same ratios (1:1, 2:1, 3:1) are used to train the data and then test the data over the complete dataset. The models used for evaluation are Logistics Regression, Random Forest and SVM.

Logistic Regression:

Logistic Regression is a type of classification algorithm involving a linear discriminant. Unlike actual regression, logistic regression does not try to predict the value of a numeric variable given a set of inputs. Instead, the output is a probability that the given input point belongs to a certain class.

The parameter tuning is not done in this case for Logistic regression but if the parameter tuning is done for Logistic regression, it will give better output than what is show below.

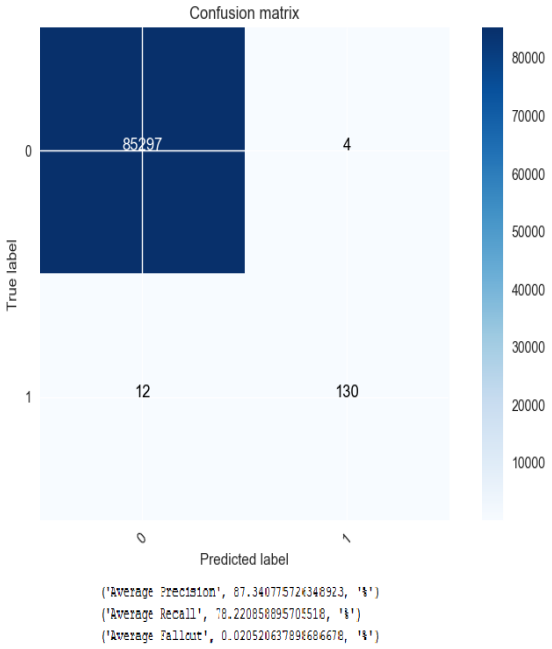
In the below graph we have trained logistic regression on the dataset to get the values of Recall and Precision.



Confusion Matrix for Logistic Regression

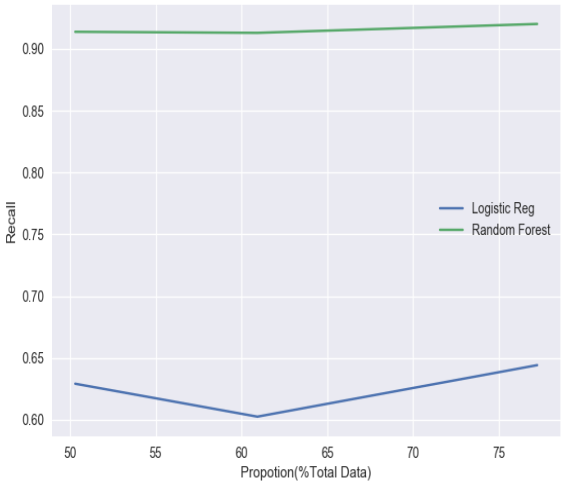
Random Forest Classifier:

A random forest is a Meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.



Confusion Matrix for Random Forest

The comparison for the two models is as below:



Model comparison for oversampling

Random Forest clearly has better recall value. However this comes at the cost of overfitting the model due to duplicating the samples.

Stratified K Fold

Since the data in the dataset is highly imbalance, it is very important to take account the ratio of fraud / non fraud classes while splitting the data in training / test set.

We have used stratified k-fold cross validation for this purpose because in this the training set is partitioned based on class, folding is performed in each partition and then the results are merged. This ensures that the number of instances for each class in each fold is approximately the same as the distribution of classes in the training set.

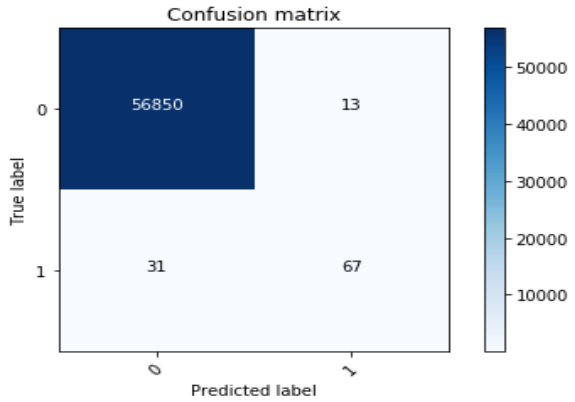
We have split the dataset into five folds, so the classifier will be trained five times. Each training data set will have different shuffled data and different test data. We are calculating precision and recall in each iteration and taking its average after all the interactions are done.

```
skf = StratifiedKFold(n_splits = 5 , shuffle = True)
for train_index, test_index in skf.split(X, y):
    X_train, y_train = X[train_index], y[train_index]
    X_test, y_test = X[test_index], y[test_index]
    lrn.fit(X_train, y_train)
    y_pred = lrn.predict(X_test)
```

We are training the data using various classifier and observing the result as below:

Logistic Regression:

In the below confusion matrix, we are getting precision value 84% and recall value is 68%. If we consider all iterations precision value varies from 93.3% to 83.8% and recall value from 57.1% to 68.4%.

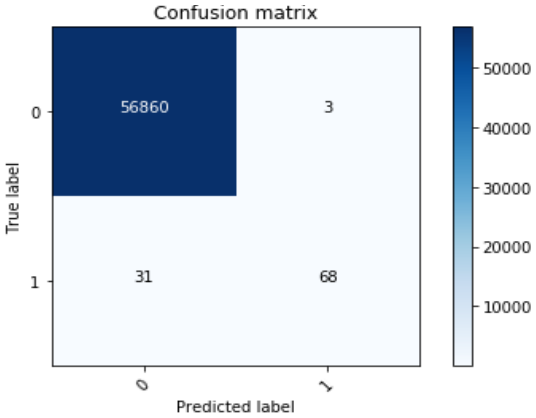



```
Precision =     0.838
Recall (TPR) =  0.684
Fallout (FPR) = 2.286e-04
Average Precision 87.2086530264 %
Average Recall 62.2057307772 %
Average Fallout 0.0161792378172 %
```

The result of this prediction is depicted in the confusion matrix. We can see that almost all non-fraudulent transactions are also recognized as such. About 2/3 of all frauds are detected, but quite many are not recognized.

Support Vector Machines:

We have trained a Support Vector Machine on the training set, and predicted the outcome on the testing set. The result of this prediction is depicted in the confusion matrix below.

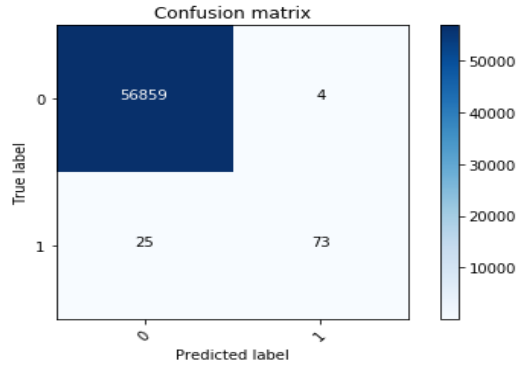



```
Precision =     0.958
Recall (TPR) =  0.687
Fallout (FPR) = 5.276e-05
```

In SVM, recall value is quite similar to logistic regression recall value however SVM precision is better than logistic regression.

Random Forest:

When we trained random forest classifier, we got much better results as compare to logistic regression and SVM. In this case precision varies from 92.4% to 96.4% and recall varies from 73.7% to 81.6%. The result from one of the iteration is shown below in confusion matrix.



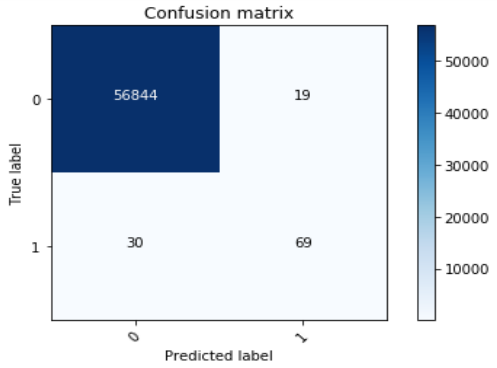

```
Precision =     0.948
Recall (TPR) =  0.745
Fallout (FPR) = 7.034e-05
```

The average precision and recall for all five iterations are as below

```
Average Precision 94.7153613825 %
Average Recall 76.6316223459 %
Average Fallout 0.00738617378612 %
```

Decision Trees

In case of decision tree classifier, recall value is better than logistic regression and SVM but precision value is very low comparatively. Recall value varies from 69.7% to 80.1% and precision value varies from 72% to 80.6%. Below is confusion matrix for one of the iterations.




```
Precision =     0.784
Recall (TPR) =  0.697
Fallout (FPR) = 3.341e-04
```

The average precision and recall for all five iterations are as below

```
Average Precision 75.9064374808 %
Average Recall 73.9868068439 %
Average Fallout 0.0407998171043 %
```

If we compare all four classifier with stratified K-fold validation than Random forest comes to be the best among four having decent precision value (around 94%) and recall value (76%).

SMOTE

SMOTE is an over-sampling approach in which the minority class is over-sampled by creating "synthetic" examples rather than by over-sampling with replacement. We generate synthetic examples in a less application-specific manner, by operating in "feature space" rather than "data space". The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. Depending upon the amount of over-sampling required, neighbors from the k nearest neighbors are randomly chosen. Our implementation currently uses five nearest neighbors. The majority class is under-sampled by randomly removing samples from the majority class population until the minority class becomes some specified percentage of the majority class. So, SMOTE technique is actually the mixture of under sampling and over sampling.

We first started with sampling of 100% dataset and implemented smote over it. As SMOTE implements synthetic creation of minority class, so having more number of minority class i.e. 100% instead of 70% can create the better samples which are much closer to the actual TP class.

So after implementing SMOTE on 100% dataset, below is the oversampled data set that we have obtained:

```
from imblearn.over_sampling import SMOTE
os = SMOTE(random_state=0)
os_data_X,os_data_y=os.fit_sample(X_data,y_data)
os_data_X = pd.DataFrame(data=os_data_X,columns = X_data.columns )
os_data_y= pd.DataFrame(data=os_data_y,columns=["Class"])
# we can Check the numbers of our data
print("length of oversampled data is ",len(os_data_X))
print("Number of normal transcation in oversampled data",len(os_data_y[
print("No.of fraud transcation",len(os_data_y[os_data_y["Class"]==1]))
print("Proportion of Normal data in oversampled data is ",len(os_data_
print("Proportion of fraud data in oversampled data is ",len(os_data_y[
```

-Length of oversampled data is: 568630

-Number of normal transactions in oversampled data :284315

-No. of fraud transactions are: 284315

-Proportion of Normal data in oversampled data is: 0.5

-Proportion of fraud data in oversampled data is: 0.5

After Oversampling, we have trained the dataset on all the mentioned classifier and the results are as below:

Logistic Regression:

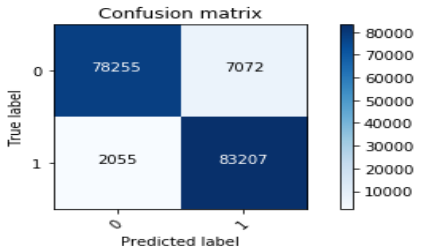
Logistic Regression is a type of classification algorithm involving a linear discriminant. Unlike actual regression, logistic regression does not try to predict the value of a numeric variable given a set of inputs. Instead, the output is a probability that the given input point belongs to a certain class. We will first train the classifier with SMOTE sampled training data and predict the outcome with testing data.

If we see in the below outcome. The Recall and Precision values are pretty high i.e. 92 and 97 percent respectively. This shows the sign of overfitting. As mentioned previously, both the values for recall and precision cannot be high as there is always a tradeoff which one has to make between recall and precision.

We will now test this data on SVM and Random Forest to see if they also overfit the data as it is done by Logistic regression. If they do, we will try to find the reason for the same.

```
# Now start modeling
clf= LogisticRegression()
# train data using oversampled data and predict for t
model(clf,data_train_X,data_test_X,data_train_y,data_
```

```
the recall for this model is : 0.917118848665
TP 78255
TN 83207
FP 2055
FN 7072
```




```
----------Classification Report----------
             precision    recall  f1-score   support

          0       0.92      0.98      0.95     85262
          1       0.97      0.92      0.94     85327

avg / total       0.95      0.95      0.95    170589
```

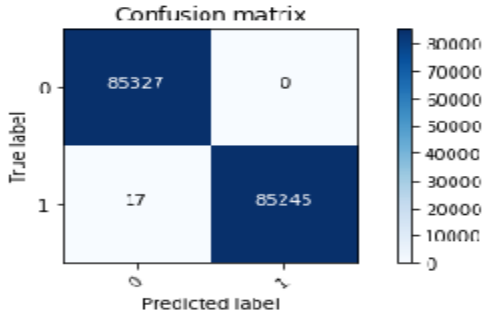
SMOTE with Logistic Regression

Support Vector Machines:

Support vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

```
# Now start modeling
clf=SVC()
# train data using oversampled data and predict for the test data
model(clf,data_train_X,data_test_X,data_train_y,data_test_y)
```

```
the recall for this model is : 0.999200413903
TP 84976
TN 85326
FP 219
FN 68
```




```
----------Classification Report----------
             precision    recall  f1-score   support

          0       1.00      1.00      1.00     85545
          1       1.00      1.00      1.00     85044

avg / total       1.00      1.00      1.00    170589
```

The above plot shows that the classifier is overfitting the dataset and therefore showing recall and precision as 1 for TP which should not be the case. We will implement the Random Forest and check if it over fits or not. If it will over fit as well, we will try to find out the reason for the same.

Random Forest Classifier:

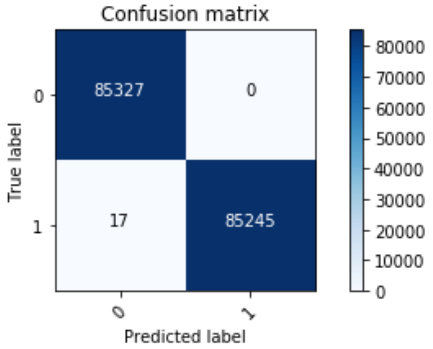
A random forest is a Meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

```
# Now start modeling
clf= RandomForestClassifier(n_estimators=100)
# train data using oversampled data and predict for the test data
model(clf,data_train_X,data_test_X,data_train_y,data_test_y)
```

The above code train the classifier on 70% data set which we have taken after splitting the oversampled data after SMOTE implementation. After testing we find that the recall and prediction values are highest i.e. 1. Which means 100 percent recall and precision.

```
# Now start modeling
clf= RandomForestClassifier(n_estimators=100)
# train data using oversampled data and predict for t
model(clf,data_train_X,data_test_X,data_train_y,data_
```

```
the recall for this model is : 1.0
TP 85327
TN 85245
FP 17
FN 0
```




```
----------Classification Report----------
             precision    recall  f1-score   support

          0       1.00      1.00      1.00     85262
          1       1.00      1.00      1.00     85327

avg / total       1.00      1.00      1.00    170589
```

The above results shows us that the data is actually overfitting. This means that the classifier has actually learned the training data and is giving perfect output for test data. The reason for this overfitting is that, the SMOTE has actually created the samples for each of the outliers as well as the regular data. This doesn't leave any of the true samples for which the samples are not generated. While doing a 70% split for whole SMOTE

implemented data, the classifier gets trained on the various samples which includes any abnormal data or the outliers. This does not leave any room for the new unexpected data which classifier can predict.

To overcome this problem of overfitting and seeing if our explanation of overfitting is actually right, we will first split the data in 70-30 ratio and then implement SMOTE on the 70 percent data. After which we will train the classifier on the same data and test it on remaining 30 percent data.

As we have seen that the Random Forest Classifier is most prone to overfitting due to the learning behavior of the classifier, we will try to implement SMOTE on 70% of data and test the remaining 30% data on the Random forest classifier. If it does not over fit Random Forest Classifier, it won't over fit the rest of the classifiers as well.

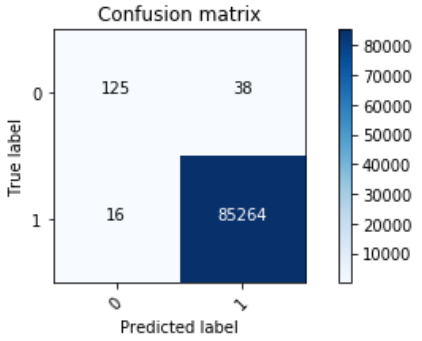
SMOTE Implementation II

As we have seen that the classifier is overfitting with the samples generated on 100% data set, we will now split the data first and then implement SMOTE on the 70 Percent dataset.

```
length of training data
199364
length of test data
85443
length of oversampled data is  398042
Number of normal transcation in oversampled data 199021
No.of fraud transcation 199021
Proportion of Normal data in oversampled data is  0.5
Proportion of fraud data in oversampled data is  0.5
```

```
# Now start modeling
clf= RandomForestClassifier(n_estimators=100)
# train data using oversampled data and predict for the
model(clf,os_data_X,data_test_X,os_data_y,data_test_y)
```

```
the recall for this model is : 0.766871165644
TP 125
TN 85264
FP 16
FN 38
```




```
----------Classification Report-----------------------
             precision    recall  f1-score   support

          0       1.00      1.00      1.00     85280
          1       0.89      0.77      0.82       163

avg / total       1.00      1.00      1.00     85443
```

The Above Scenario shows us that when we generate the SMOTE samples on 70% of data and then test on the remaining 30% data, we will have better training of classifier and it wont over fit. The recall and precision values are the best which we can get. In particular, any small improvement of recall comes with a precipitous drop of precision. This showed that the SMOTE is the best technique among the all sampling techniques if implemented after splitting the dataset.

VI. Conclusions Drawn

We have observed that the recall pattern for under sampling and oversampling for this type of data set is pretty high but the precision is pretty low. Precision is less, means we are predicting other class wrongly. As for one of the parts, there were 953 transaction which were predicted as fraud. Now High Recall and low Precision means we are catching fraud transaction very well but we are also catching innocent transaction i.e which are not fraud. If we go by that model then we are going to put 953 innocents in jail with the all criminal who have actually innocent. So with recall our precision should also be better.

After implementing techniques like stratified K fold and SMOTE, we see that the both values of recall and precision now remain close to each other and there is not vast difference between them. If parameter tuning is done for K stratified, it will perform much better than undersampled and oversampled data with good precision and recall values. As we have seen in the last that SMOTE is the best technique which gives us precision and recall of around 82-87 percent. The best classifier that worked best with SMOTE is Random Forest.

After all this analysis we have come to a conclusion that for fraud detection of transactions The final choice depends on practical considerations, like what are the costs for the bank of a false alert vs a not detected fraud.It all depends on the bank how they would pick precision and recall in practice - there will always be a tradeoff.

References

[1]"Credit Card Fraud Detection | Kaggle", *Kaggle.com*, 2017. [Online]. Available: https://www.kaggle.com/gargmanish/d/dalpozz/creditcardfraud/how-to-handle-imbalance-data-study-in-detail. [Accessed: 04- Apr- 2017].

[2]"Credit Card Fraud Detection | Kaggle", *Kaggle.com*, 2017. [Online]. Available: https://www.kaggle.com/dstuerzer/d/dalpozz/creditcardfraud/optimized-logistic-regression. [Accessed: 04- Apr- 2017].

[3]"Types of Credit Card Fraud", *People.exeter.ac.uk*, 2017. [Online]. Available: http://people.exeter.ac.uk/watupman/undergrad/owsylves/page3.html. [Accessed: 04- Apr- 2017].

[4]"EZProxy - Electronic Resource Login", *Ac.els-cdn.com.ezproxy.library.uvic.ca*, 2017. [Online]. Available: http://ac.els-cdn.com.ezproxy.library.uvic.ca. [Accessed: 04- Apr- 2017].

[5]"Data analysis techniques for fraud detection", *En.wikipedia.org*, 2017. [Online]. Available: