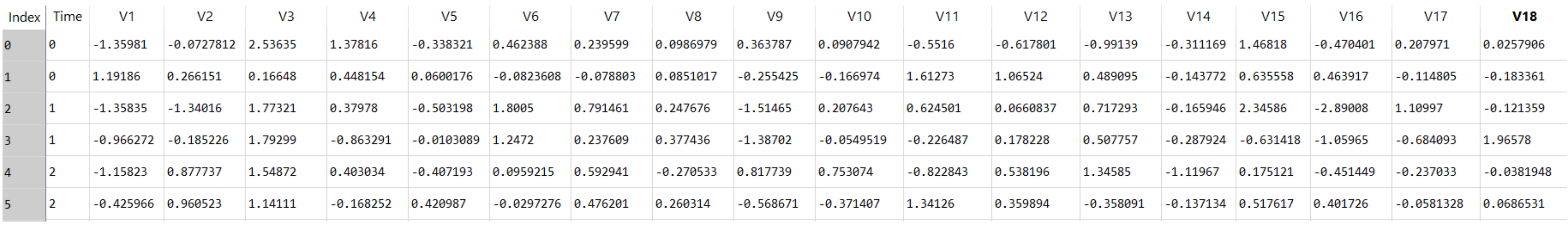
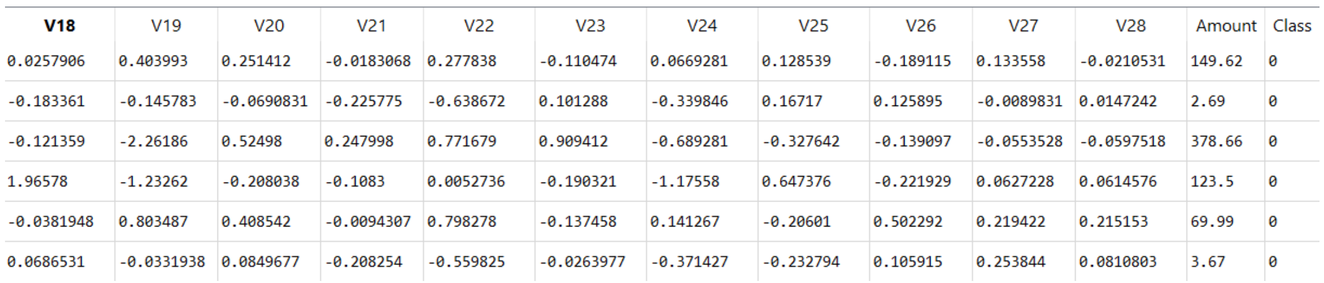
Credit Card Fraud Detection

Description of the data:

The data was taken by UCI Machine Learning repository. All the features were the results of Principle Component analysis and were unnamed due to policy considerations. Here is a snap of the data. The data consists of



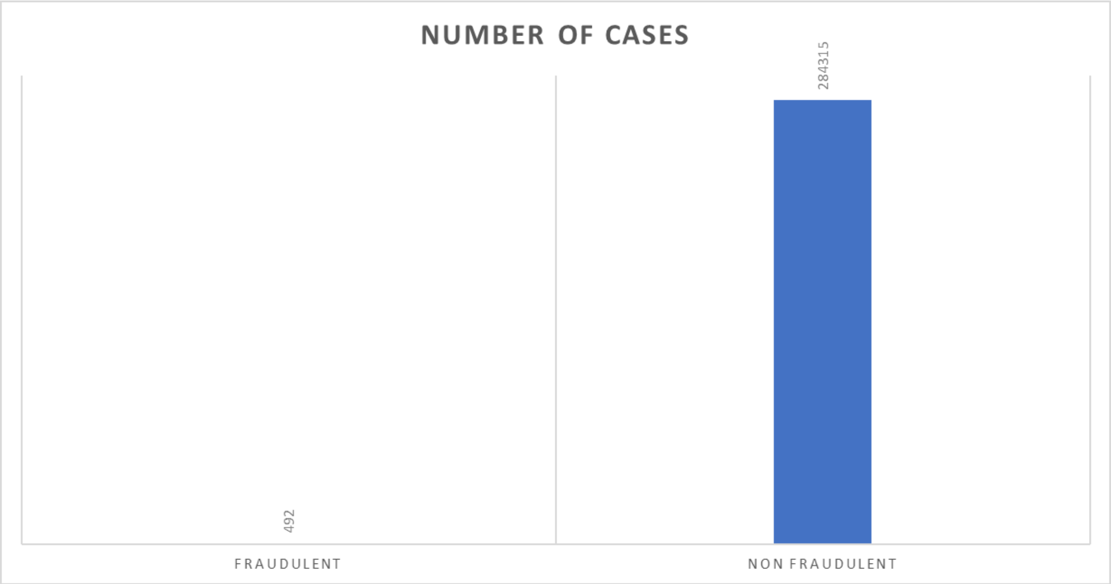
Index	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	**V18**
0	0	-1.35981	-0.0727812	2.53635	1.37816	-0.338321	0.462388	0.239599	0.0986979	0.363787	0.0907942	-0.5516	-0.617801	-0.99139	-0.311169	1.46818	-0.470401	0.207971	0.0257906
1	0	1.19186	0.266151	0.16648	0.448154	0.0600176	-0.0823608	-0.078803	0.0851017	-0.255425	-0.166974	1.61273	1.06524	0.489095	-0.143772	0.635558	0.463917	-0.114805	-0.183361
2	1	-1.35835	-1.34016	1.77321	0.37978	-0.503198	1.8005	0.791461	0.247676	-1.51465	0.207643	0.624501	0.0660837	0.717293	-0.165946	2.34586	-2.89008	1.10997	-0.121359
3	1	-0.966272	-0.185226	1.79299	-0.863291	-0.0103089	1.2472	0.237609	0.377436	-1.38702	-0.0549519	-0.226487	0.178228	0.507757	-0.287924	-0.631418	-1.05965	-0.684093	1.96578
4	2	-1.15823	0.877737	1.54872	0.403034	-0.407193	0.0959215	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196	1.34585	-1.11967	0.175121	-0.451449	-0.237033	-0.0381948
5	2	-0.425966	0.960523	1.14111	-0.168252	0.420987	-0.0297276	0.476201	0.260314	-0.568671	-0.371407	1.34126	0.359894	-0.358091	-0.137134	0.517617	0.401726	-0.0581328	0.0686531



V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0.0257906	0.403993	0.251412	-0.0183068	0.277838	-0.110474	0.0669281	0.128539	-0.189115	0.133558	-0.0210531	149.62	0
-0.183361	-0.145783	-0.0690831	-0.225775	-0.638672	0.101288	-0.339846	0.16717	0.125895	-0.0089831	0.0147242	2.69	0
-0.121359	-2.26186	0.52498	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.0553528	-0.0597518	378.66	0
1.96578	-1.23262	-0.208038	-0.1083	0.0052736	-0.190321	-1.17558	0.647376	-0.221929	0.0627228	0.0614576	123.5	0
-0.0381948	0.803487	0.408542	-0.0094307	0.798278	-0.137458	0.141267	-0.20601	0.502292	0.219422	0.215153	69.99	0
0.0686531	-0.0331938	0.0849677	-0.208254	-0.559825	-0.0263977	-0.371427	-0.232794	0.105915	0.253844	0.0810803	3.67	0

Understanding the data:

1. Minimum amount of data cleaning was required as there was no null value and the data available was a result of principle component analysis.
2. The data is highly imbalanced with 0.17% fraudulent cases and 99.83% of non-fraudulent cases.



Problems with imbalanced data set

1. Our model will be highly biased towards the non-fraudulent class, or the model will be overfitted to one class.
2. We will draw incorrect correlation between the features with imbalanced data set.

Metric of evaluation:

- Accuracy will not be a good metric here, because even if our model predicts the transaction to be non-fraudulent all the times, still it is 99.87% accurate
- Not all errors made by the model have same weight, for example, error in classifying a fraudulent transaction to be non-fraudulent is much more severe than classifying a fraudulent transaction as fraudulent.
- An appropriate metric here will be the true positive rate a.k.a. recall, which is **recall** is the number of true positives divided by the number of true positives plus the number of false negatives.

What can be done to tackle this problem?

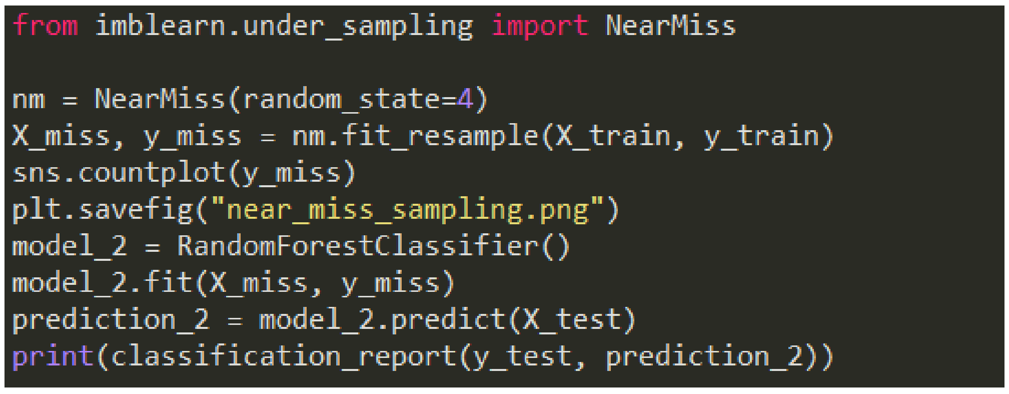
- Minority class can be over sampled to match the number of majority class. This can be done by simulating artificial samples from minority class by K-Nears neighbour algorithm
- On the similar grounds, non-fraudulent transactions can be under sampled to be have it’s count equal to the minority class
- For incorporating the difference in weights of errors, we can plot ROC curve and estimate a best threshold value for classification of classes based on the needs of company. This threshold can be decreased if they don’t want to take any chances of leaving out fraudulent transaction on the expense of more false positives, or we can increase this threshold to have more accurate predictions on the cost of more false negatives.

Sampling of data:

- Oversampling using Synthetic Minority Over Sampling Technique (SMOTE) and SMOTE Tomek using the ratio 0.1
- Under sampling using near Miss
- Comparing the results obtained from SMOTE, SMOTE Tomek and Near Miss

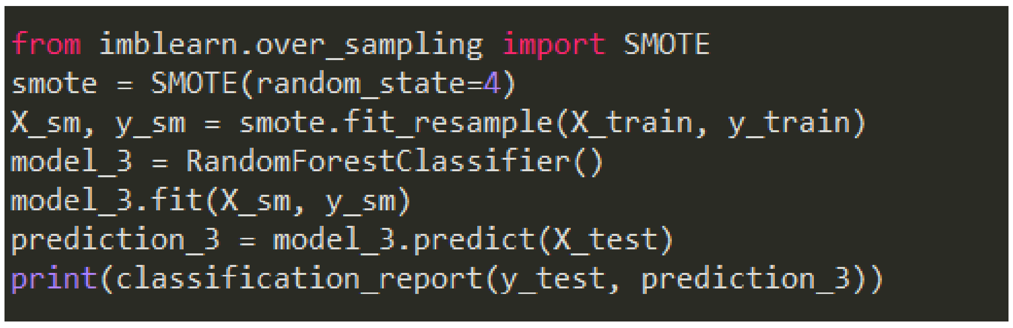
The draw backs which each kind of sampling is that, we will have information loss in the case of undersampling, and our model may learn from false trends and be biased towards it, in the case of oversampling.

Implementing Under Sampling



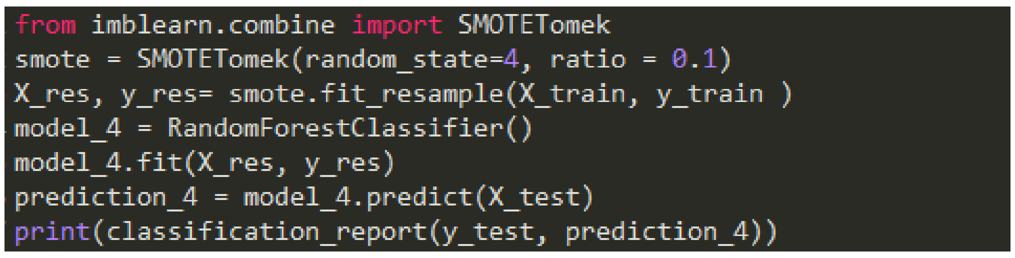
```
from imblearn.under_sampling import NearMiss

nm = NearMiss(random_state=4)
X_miss, y_miss = nm.fit_resample(X_train, y_train)
sns.countplot(y_miss)
plt.savefig("near_miss_sampling.png")
model_2 = RandomForestClassifier()
model_2.fit(X_miss, y_miss)
prediction_2 = model_2.predict(X_test)
print(classification_report(y_test, prediction_2))
```

Implementing Over Sampling



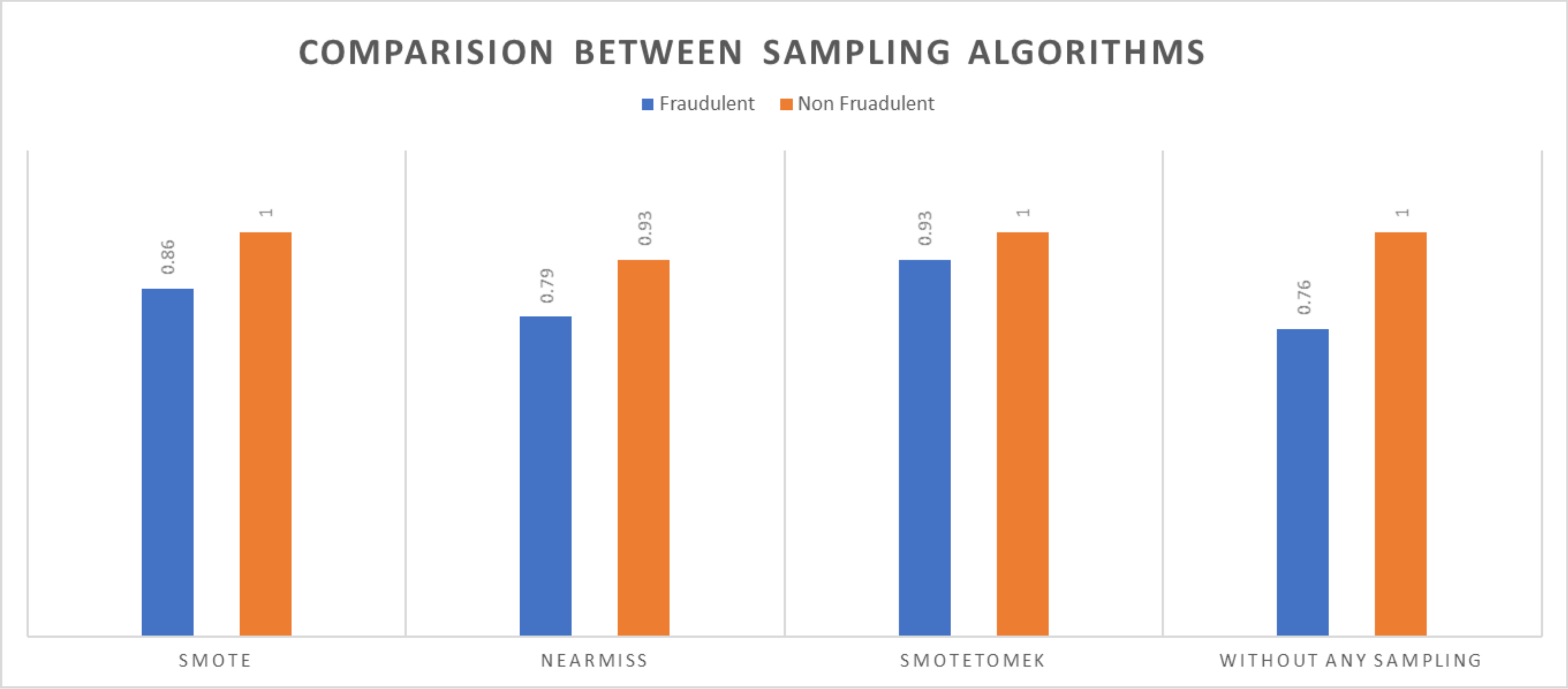
```
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=4)
X_sm, y_sm = smote.fit_resample(X_train, y_train)
model_3 = RandomForestClassifier()
model_3.fit(X_sm, y_sm)
prediction_3 = model_3.predict(X_test)
print(classification_report(y_test, prediction_3))
```

A hybrid sampling, SMOTE Tomek



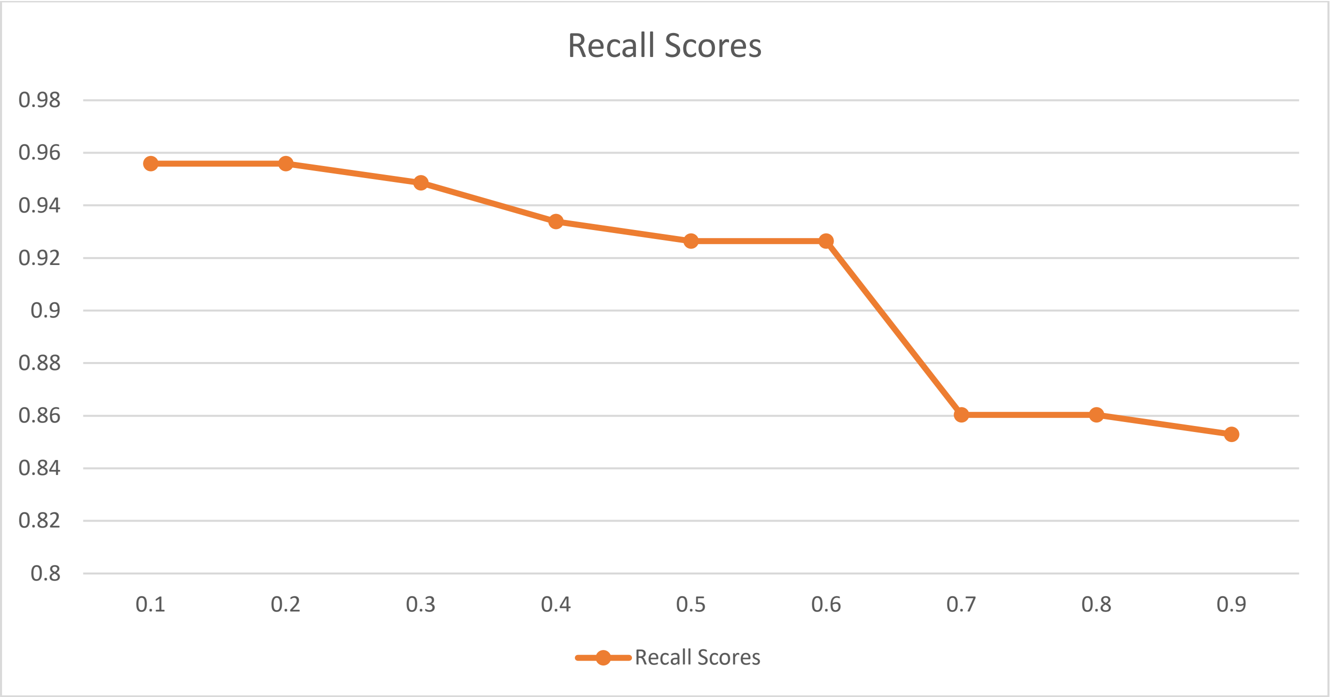
```
from imblearn.combine import SMOTETomek
smote = SMOTETomek(random_state=4, ratio = 0.1)
X_res, y_res= smote.fit_resample(X_train, y_train )
model_4 = RandomForestClassifier()
model_4.fit(X_res, y_res)
prediction_4 = model_4.predict(X_test)
print(classification_report(y_test, prediction_4))
```

Comparing the sampling methods



- Our metric for judgement here is recall instead of accuracy, because identifying the true positives in this case is much more important than identifying the true negatives.
- We are interested in finding the recall on the fraudulent class
- Without any sampling, we will be having a 0.76 recall, which goes up to 0.93 in the case of SMOTE Tomek, this shows the impact of imbalanced data sets on the result
- As speculated, the hybrid sampling performs the best here

Choosing appropriate Threshold for our Random Forest Algorithm



The best recall score is obtained with thresholds 0.1 and 0.2, however a threshold of 0.1 improves the recall at the cost of increasing false positives, hence a threshold of 0.2 is considered.

So now we have a model that predicts the fraudulent transactions, to make this model more viable, I have implemented it in excel by using python as a backend. This is done using xlwings plugin that allows to use python scripts in Microsoft Excel.

A function was written which takes in three argument:

1. Range of the input data
2. Number of test samples
3. The output cells

It runs python script in the backend and outputs the predicted results in the excel sheet.