CHAPTER 5
IMPLEMENTATION AND TESTING

5.1 Implementation

In the csv dataset, there are several columns to be labeled separately. However, there exist 2 column which contains the same data that is cell number and account number. Account number is the cell number used to register an account. These 2 columns cannot be encoded seperately, as same cell number and account number is an important factor to determine if a certain transaction is fraudulent.

```
# to count unique values in account & cell number
cell_labels = []

    for obj in df['cell_number']:
      if obj not in cell_labels:
        cell_labels.append(obj)

    for obj in df['account_number']:
      if obj not in cell_labels:
        cell_labels.append(obj)

index = 0

# assign number to unique cell number in the form of index
for obj in df['cell_number']:
    result = cell_labels.index(obj)
    df.at[index,'cell_number'] = result
    index = index + 1

index = 0

# assign number to unique account number which is often a cell number in the form of index
for obj in df['account_number']:
    result = cell_labels.index(obj)
    df.at[index,'account_number'] = result
    index = index + 1
```

It’s tricky to label encode 2 columns since the only available library for encoding only take 1 column to encode, while in this scenario 2 different column needs to be encoded at the same time so identical number can be spotted easily. The researcher wrote the above line specifically to do the task.

The employer suggest that the program should be able to tell the user which transaction is predicted correctly. Below is piece of code designed to locate which data is identified as false positives and false negatives which is originally there for testing but is also important to the clients to know just which transaction is prone to misclassified

```
# print(train_features.loc[1])
# print('Accuracy : ',rfacc,'%')
# print('TP : ',TP)
# print('TN : ',TN)
# print('FP : ',FP)
# print(test_features.loc[(test_labels == 0) & (predictions == 1)])
# print('FN : ',FN)
# print(test_features.loc[(test_labels == 1) & (predictions == 0)])
# print("F1 score: {:.2f}".format(score))
```

5.2 Testing

To test the algorithm, the researcher runs the each algorithm 30x and records each True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN), count the precision and recall to determine which algorithm is more suitable to be used.

Table 5.1: Scenario 1 result

SC1	TP	TN	FP	FN	PREC	RECA
RF	40.06666667	32565.13333	4	0.066666667	0.909228442	0.99833887
SVM	19.06666667	32543.06667	26.06666667	21.06666667	0.422451994	0.475083056
SVM GAUSSIAN	1.066666667	32569.06667	0.066666667	39.06666667	0.941176471	0.026578073
RF + PCA	22.3	32569.06667	0.066666667	17.83333333	0.997019374	0.555647841

Table 5.2: Scenario 2 result

SC2	TP	TN	FP	FN	PREC	RECA
RF	38	32569.06667	0.066666667	2.133333333	0.998248687	0.946843854
SVM	13.06666667	32558.06667	11.06666667	27.06666667	0.541436464	0.325581395
SVM GAUSSIAN	0.066666667	32569.06667	0.066666667	40.06666667	0.5	0.00166113
RF + PCA	26.63333333	32557.96667	11.16666667	13.5	0.704585538	0.663621262

Table 5.3: Scenario 3 result

SC3	TP	TN	FP	FN	PREC	RECA
RF	40.06666667	32560.06667	9.066666667	0.066666667	0.815468114	0.99833887
SVM	0.066666667	32569.06667	0.066666667	40.06666667	0.5	0.00166113
SVM GAUSSIAN	8.066666667	32569.06667	0.066666667	32.06666667	0.991803279	0.200996678
RF + PCA	31.73333333	32567.93333	1.2	8.4	0.963562753	0.790697674

Table 5.4: Scenario 4 result

SC4	TP	TN	FP	FN	PREC	RECA
RF	37.36666667	32569.06667	0.066666667	2.766666667	0.998219056	0.931063123
SVM	0.066666667	32545.06667	24.06666667	40.06666667	0.002762431	0.00166113
SVM GAUSSIAN	1.066666667	32569.06667	0.066666667	39.06666667	0.941176471	0.026578073
RF + PCA	15.96666667	32569.06667	0.066666667	24.16666667	0.995841996	0.397840532

Table 5.5: Scenario 5 result

SC5	TP	TN	FP	FN	PREC	RECA
RF	38.06666667	32569.06667	0.066666667	2.066666667	0.998251748	0.948504983
SVM	0.066666667	32565.06667	4.066666667	40.06666667	0.016129032	0.00166113
SVM GAUSSIAN	0.066666667	32569.06667	0.066666667	40.06666667	0.5	0.00166113
RF + PCA	27.13333333	32561.06667	8.066666667	13	0.770833333	0.676079734

Table 5.6: Scenario 6 result

SC6	TP	TN	FP	FN	PREC	RECA
RF	29.53333333	32569.06667	0.066666667	10.6	0.997747748	0.735880399
SVM	0.066666667	32569.06667	0.066666667	40.06666667	0.5	0.00166113
SVM GAUSSIAN	0.066666667	32569.06667	0.066666667	40.06666667	0.5	0.00166113
RF + PCA	20.83333333	32569.06667	0.066666667	19.3	0.996810207	0.51910299

Table 5.7: Scenario 7 result

SC7	TP	TN	FP	FN	PREC	RECA
RF	40.06666667	32559.06667	10.06666667	0.066666667	0.799202128	0.99833887
SVM	0.066666667	32569.06667	0.066666667	40.06666667	0.5	0.00166113
SVM GAUSSIAN	8.066666667	32569.06667	0.066666667	32.06666667	0.991803279	0.200996678
RF + PCA	30.96666667	32568.96667	0.166666667	9.166666667	0.994646681	0.771594684

Table 5.8: Overall Result

Scenarios	RF		SVM		SVM GAUSSIAN		RF PCA	
	PREC	RECA	PREC	RECA	PREC	RECA	PREC	RECA
1	90.9%	99.8%	42.2%	47.5%	94.1%	2.7%	99.7%	55.6%
2	99.8%	94.7%	54.1%	32.6%	50%	0.2%	70.5%	66.4%
3	81.5%	99.8%	50%	0.2%	99.2%	20.1%	96.4%	79.1%
4	99.8%	93.1%	0.3%	0.2%	94.1%	2.7%	99.6%	39.8%
5	99.8%	94.9%	1.6%	0.2%	50%	0.2%	77.1%	67.6%
6	99.8%	73.6%	50%	0.2%	50%	0.2%	99.7%	51.9%
7	79.9%	99.8%	50%	0.2%	99.5%	20.1%	99.5%	77.2%

The author uses precision and recall to determine the methods accuracy and if it's safe to be used in real-world scenario. Precision refers to the method's reliability when it detects fraudulent transaction, precision is the portion of correct fraud classification from all fraud classification the algorithm made; when the algorithm has high precision, we can trust it when it classifies a transaction as fraudulent. Recall is the method's accuracy when it identifies a fraudulent transaction, recall is the portion of correct fraudulent transaction prediction from all fraudulent transactions in the dataset; high recall score means it correctly identifies the majority of fraudulent transactions.

With imbalanced dataset these 2 scores is the most reliable method of determining the algorithms performance, since the portion of legitimate transaction is overwhelmingly larger than the fraudulent transaction, a model that just classify everything as legitimate transaction will net about 98% accuracy since it does not calculate the weight of misclassify the minority class which is small in proportion but is very important. If an algorithm has high precision but low recall, we can see that every time the algorithm classifies a transaction as fraudulent, we can trust that, but there are many false negatives or many fraudulent transactions classified as legitimate transaction. Conversely with low precision but high recall, we'll see an increase in the false positives, the algorithm correctly classifies the majority of fraudulent transaction but misclassify some legitimate transaction as fraudulent.

In all of the scenarios above, SVM with RBF kernel almost always have higher precision but lower recall than SVM linear, interestingly, the scenarios where SVM RBF's precision dropped is when the phone number is being randomly generated so there's no duplicate. Thus, we can conclude that SVM RBF relies on phone number greatly to predict fraudulent transactions. Across all scenario, SVM Linear precision is capped at 50% and drops sharply on scenario 4 and 5 which have nothing in common, and maintains a low recall on every scenario bar the first two which has date and product type. Thus, we can conclude that SVM relies more on the date related feature and product type to makes prediction, but fails to predict reliably since there's so many false positives and negatives.

RF with PCA applied have higher precision and recall than SVMs, maintaining real perfect precision on almost every scenario and experiences a drop in scenarios where no phone number is duplicated which is 2 and 5 but interestingly not 6 which has date feature yet not 7 where date feature is removed. The author hypothesized that this algorithm relies in phone number mainly, but can switch to other feature when necessary, yet may still experience a drop in prediction quality. RF PCA has a large margin of recall score ranging from 39% to 70% which is too large to be used reliably. Default RF algorithm consistently nets very high precision and recall with a drop in score to 70% only in scenario 6 and 7 in which it has 70% recall and 77% precision respectively.

Aside from primary test to obtain the scores above, the author also has run several tests to shed light on why the algorithm have high score or lower score than other algorithm and behaves as such. SVM with linear kernel when applied to this particular dataset which has a lot of negative class, will undoubtedly misclassify some positives class as the negative class. This is what causes the many false negative in the SVM algorithm. Similar situation applies to SVM with RBF kernel which, while having slightly lower false positives in all but 2 scenarios, still biased toward the majority class. From this we can assume SVM with linear kernel, when given imbalanced dataset will perform poorly and is biased toward the majority class.

PCA calculates the effectiveness of a feature to another feature and ignores the ineffective feature in the favor of effective feature, this may be a problem if the feature ignored carries some weight in it however small. Since RF with PCA consistently has lower precision and recall score than unmodified RF except in scenario 1 and 3 which PCA has higher precision by 9% and 15% respectively even though the RF algorithm itself is exactly the same, the author concludes that PCA is unnecessary when using RF algorithm.