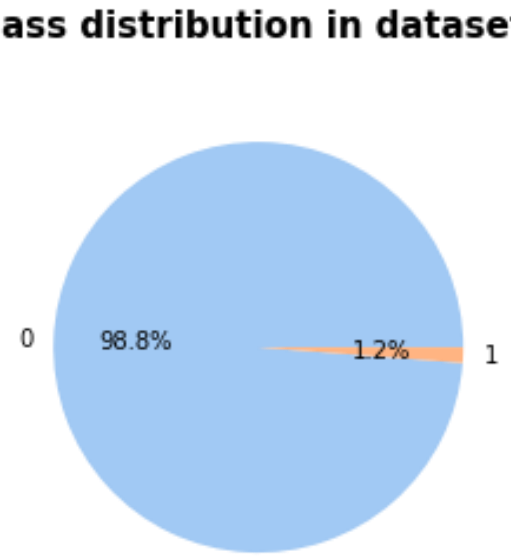
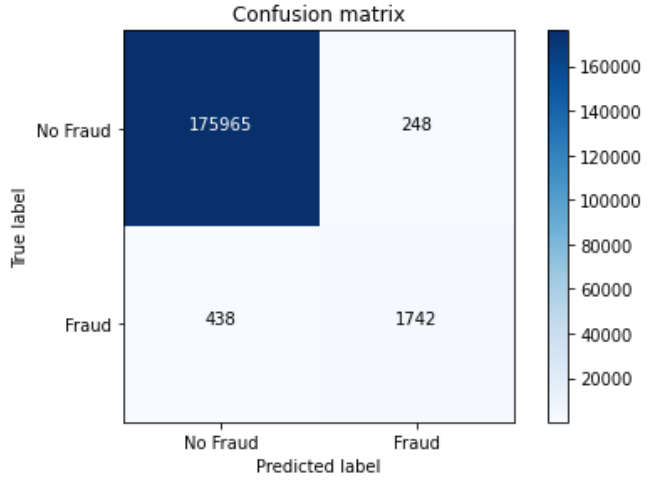
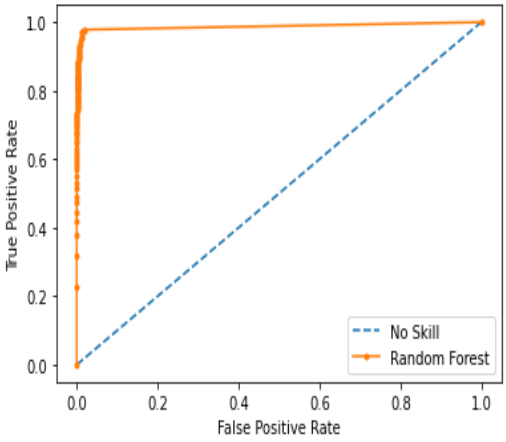
Important Experiment Results and Visualizations

Implementation without handling class imbalance issue:



Random Forest:

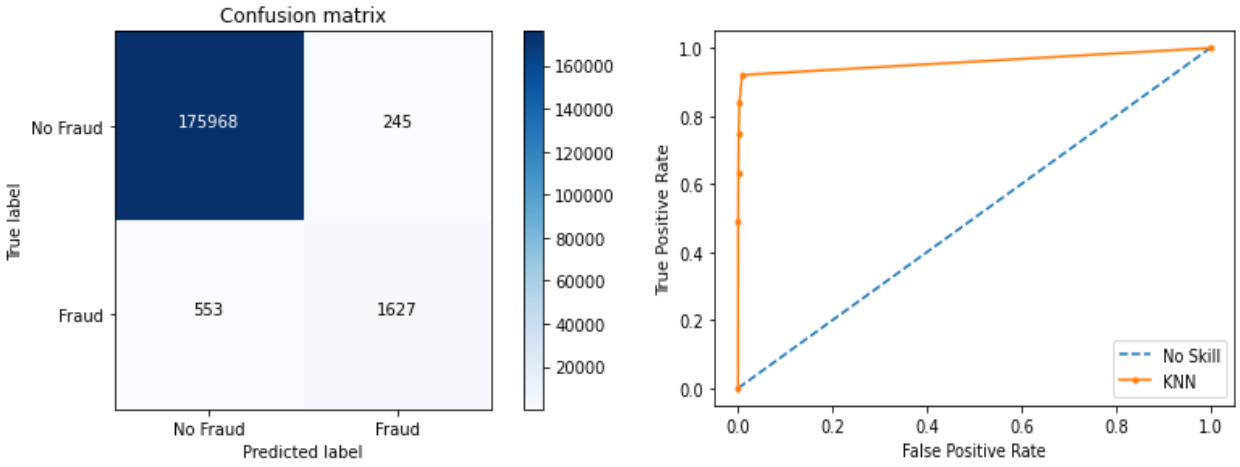




	precision	recall	f1-score	support
0	1.00	1.00	1.00	176213
1	0.88	0.80	0.84	2180
accuracy			1.00	178393
macro avg	0.94	0.90	0.92	178393
weighted avg	1.00	1.00	1.00	178393

Wall time: 5min 53s

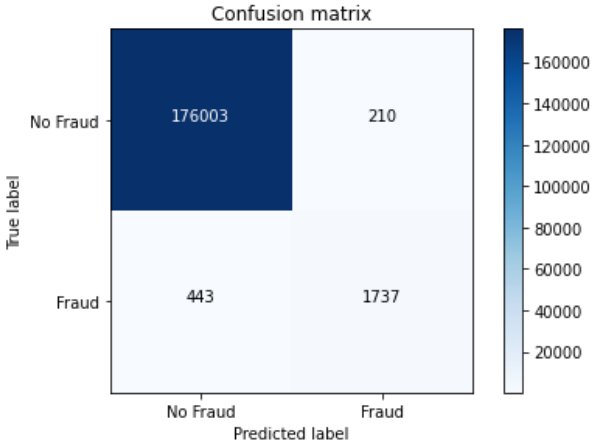
KNN:

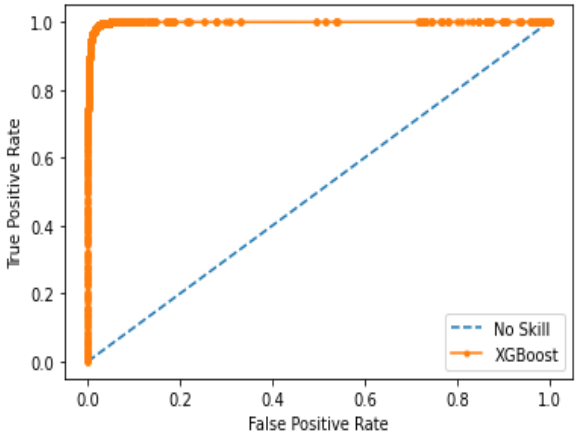


	precision	recall	f1-score	support
0	1.00	1.00	1.00	176213
1	0.87	0.75	0.80	2180
accuracy			1.00	178393
macro avg	0.93	0.87	0.90	178393
weighted avg	1.00	1.00	1.00	178393

Wall time: 3h 30min 12s

XGBoost:

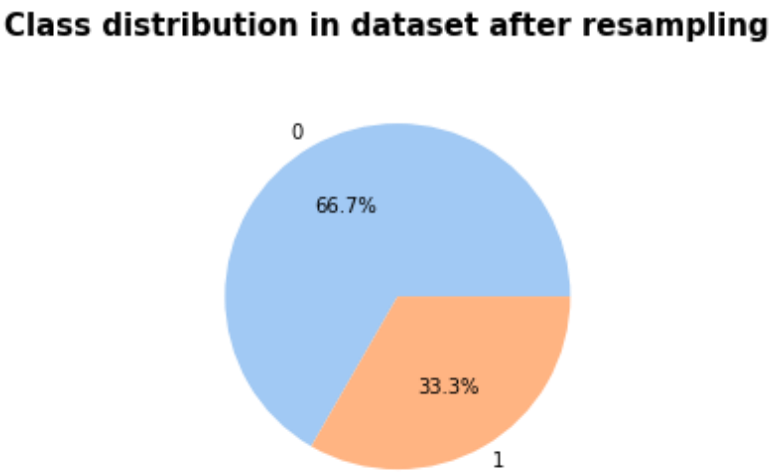




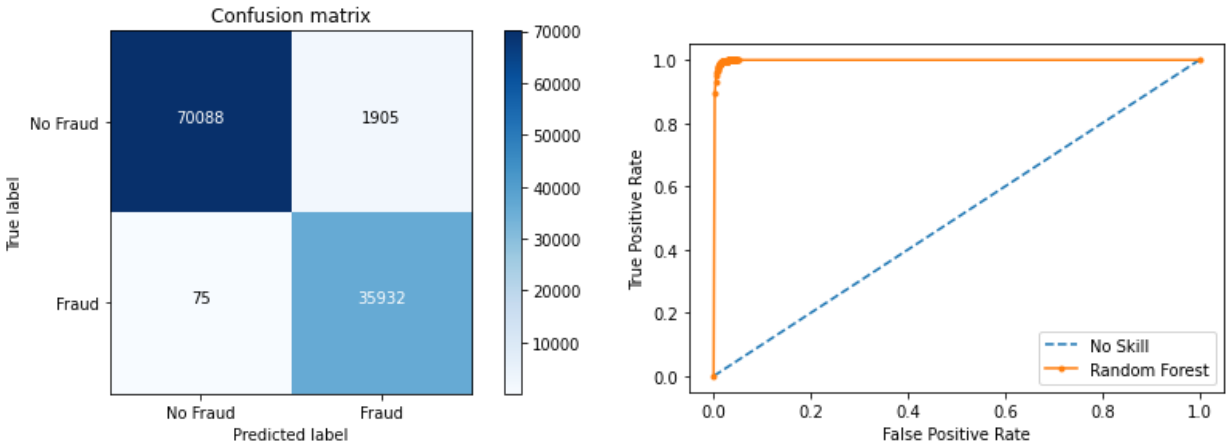
	precision	recall	f1-score	support
0	1.00	1.00	1.00	176213
1	0.89	0.80	0.84	2180
accuracy			1.00	178393
macro avg	0.94	0.90	0.92	178393
weighted avg	1.00	1.00	1.00	178393

Wall time: 12min 42s

Implementation after handling class Imbalance issue:



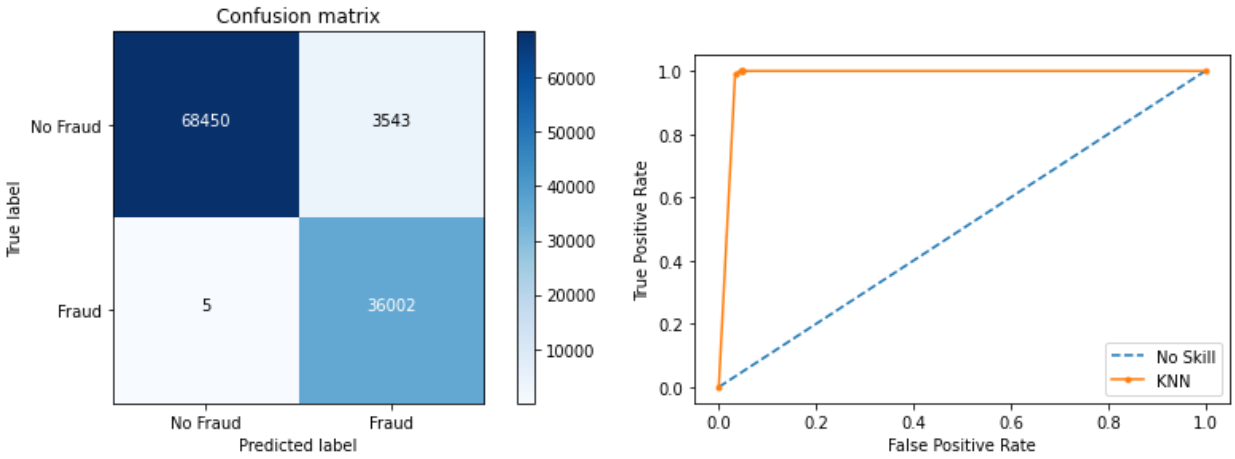
Random Forest



	precision	recall	f1-score	support
0	1.00	0.97	0.99	71993
1	0.95	1.00	0.97	36007
accuracy			0.98	108000
macro avg	0.97	0.99	0.98	108000
weighted avg	0.98	0.98	0.98	108000

Wall time: 4min 35s

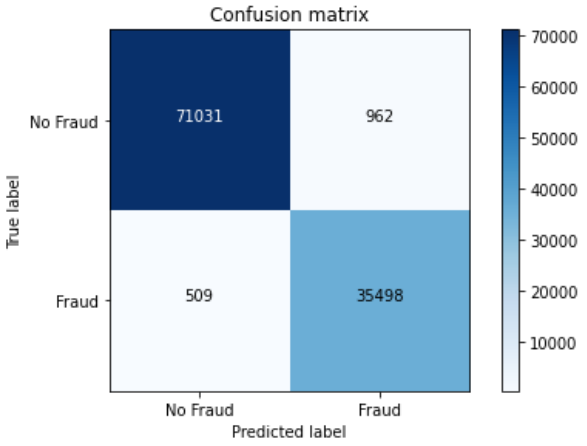
KNN

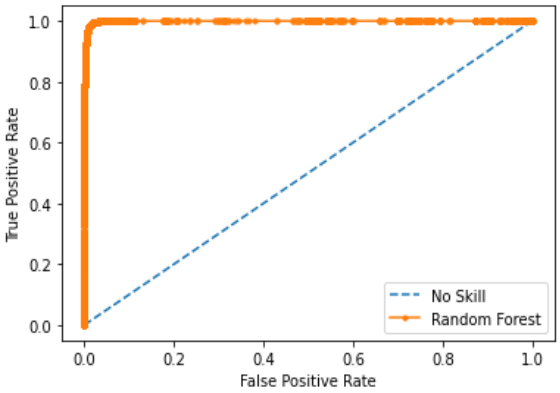


	precision	recall	f1-score	support
0	1.00	0.95	0.97	71993
1	0.91	1.00	0.95	36007
accuracy			0.97	108000
macro avg	0.96	0.98	0.96	108000
weighted avg	0.97	0.97	0.97	108000

Wall time: 29min 13s

XGBoost





	precision	recall	f1-score	support
0	0.99	0.99	0.99	71993
1	0.97	0.99	0.98	36007
accuracy			0.99	108000
macro avg	0.98	0.99	0.98	108000
weighted avg	0.99	0.99	0.99	108000

Wall time: 7min 34s

Testing the models with sample data:

customer	merchant	category	amount	fraud	custPageRank	merchPageRank	merchDegree	custDegree	merchCommunity	custCommunity
'C583110837'	'M480139044'	'es_health'	0.005313	1	0.0	0.298948	0.336475	0.448276	608504	608498
'C1093826151'	'M348934600'	'es_transportation'	0.000546	0	0.0	1.000000	1.000000	0.172414	608498	608498

[77]:
```
## testing using different models. 0-->no fraud  1-->fraud

predicted_results = {}
predicted_results['random forest'] = rf.predict(test_data)
predicted_results['KNN'] = knn.predict(test_data)
predicted_results['XGBoost'] = xgb.predict(test_data)

print(predicted_results)
```

```
{'random forest': array([1, 0], dtype=int64), 'KNN': array([1, 0], dtype=int64), 'XGBoost': array([1, 0])}
```