Project 3: Credit card fraud detection using both Classification & Clustering techniques.

Problem Statement:

The datasets contains transactions made by credit cards in September 2013 by european cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions. Goal is to perform credit card fraud detection using both Classification and clustering models.

Approach:

1. Exploratory data analysis: Gathered statistics info related to data, plotted histograms to see the imbalance in the classes, relationship between the time of the day and the transaction label. Summary on data analysis:
   a. The transaction amount is relatively small. The mean of all the mounts made is approximately USD 88.
   b. There are no "Null" values, so we don't have to work on ways to replace values.
   c. Most of the transactions were Non-Fraud (99.83%) of the time, while Fraud transactions occurs (0.17%) of the time in the dataframe.
   d. Since the initial dataset is highly imbalanced(positive class is less than 0.17 %), we ran th e tests on 4 types of data.
      1) As-is without making any changes to the given dataset.
      2) Under-sample the highly represented class(i.e; negative class/non-fraud transactions in this example)
      3) Over-sample the under represented class(i.e; positive class/fraud transactions)
      4) SMOTE(Synthetic Minority Over-sampling Technique) – SMOTE creates new synthetic points for the underrepresented class in order to have an equal balance of the classes. It uses the k nearest neighbors logic to create the new data. This is another alternative for solving the "class imbalance problems".
2. Data preprocessing: Cyclical/Sinusoidal encoding for 'time' attribute
3. Split the data into train and test set using scikit-learn library's train_test_split method. Test set size is 25% of the total dataset.
4. Machine Learning Models:
   a. We will then compare what happens when using resampling and when not using it. We will test this approach using the below classifiers,
      i. Logistic regression
      ii. RandomForestClassifier
      iii. KNeighborsClassifier
      iv. SVM
      v. MLP
   b. We will evaluate the models by using some of the performance metrics - f1-score, precision, recall and average precision score(APS)
   c. Try Undersampling and Oversampling approaches and get the performance metrics again.
   d. Try the clustering algorithms – kmeans and AGNEST on the given dataset.
   e. Compare the performance of Classification and Clustering algorithms.

Experiments and results: Below data is for the test set.

Classification:

Data	LogisticRegression					RandomForestClassifier					KNeighborsClassifier					SVM			
	f1	prec	recall	acc	APS	f1	prec	recall	acc	APS	f1	prec	recall	acc	APS	f1	prec	recall	acc
As-is	0.7444	0.9121	0.6288	0.9992	0.5742	0.8880	0.9817	0.8106	0.9996	0.7961	0.7465	0.9529	0.6136	0.9992	0.5855	0.4973	0.8679	0.3485	0.9987
Under-sample	0.0830	0.0436	0.8864	0.9637	0.0388	0.1603	0.0883	0.8712	0.9831	0.0771	0.0825	0.0433	0.8636	0.9644	0.0377	0.0319	0.0163	0.7348	0.9173
Over-sample	0.1252	0.0674	0.8788	0.9772	0.0595	**0.8889**	**0.9730**	**0.8182**	**0.9996**	**0.7964**	0.8083	0.8981	0.7348	0.9994	0.6605	0.2106	0.1201	0.8561	0.9881
SMOTE	0.1559	0.0855	0.8788	0.9824	0.0754	0.8750	0.9032	0.8485	0.9996	0.7667	0.7129	0.6316	0.8182	0.9988	0.5171	0.2414	0.1407	0.8485	0.9901

Clustering:

Data	KMeans					AGNEST				
	f1	prec	recall	acc	APS	f1	prec	recall	acc	APS
As-is	0.006	0.003	0.037	0.980	0.002	0.658	0.496	0.978	0.493	0.496
Under-sample	0.135	0.649	0.075	0.517	0.511	0.660	0.498	0.978	0.497	0.498
Over-sample	0.128	0.678	0.071	0.519	0.513	0.015	0.583	0.008	0.501	0.501
SMOTE	0.132	0.680	0.073	0.519	0.513	0.004	0.002	1.000	0.020	0.002

Conclusions:

Overall classification models performed better than the clustering models. With imbalanced datasets, accuracy is not the most appropriate way to judge the model. It is usually high and misleading. Hence we will be using the f1-score, precision, recall and average precision score.
Scenario 1(As-is data): RandomForestClassifier seems to be best on basis of all scoring parameters.
Scenario 2(Under-sample): Under sampling of over represented class(non fraud) didn't seen to be a good approach. Scores were low for all the classifiers.
Scenario 3(Over-sample): RandomForestClassifier seems to be best on basis of all scoring parameters.
Scenario 4(Over-sample using SMOTE): RandomForestClassifier seems to be best on basis of all scoring parameters.
RandomForest Classifier with over-sampling of the under-represented class(fraud transactions) seems to be the best model with f1-score = 0.8889, precision = 0.9730, recall = 0.8182, accuracy = 0.9996 and average_precison_score = 0.7964

Few observations:

1. When scaled between 0 to +1 / -1 to +1, the models are not working better in any of the scenario.
2. Model work the best when the features are not scaled. Tried MinMaxScaler, RobustScaler and StandardScaler.
3. Increasing the number of trees in RandomForest from the default 10 to 100 doesn't increase the score substantially.

References:

1. Dataset - https://www.kaggle.com/mlg-ulb/creditcardfraud
2. https://towardsdatascience.com/detecting-credit-card-fraud-using-machine-learning-a3d83423d3b8
3. https://machinelearningmastery.com/imbalanced-classification-with-the-fraudulent-credit-card-transactions-dataset/

4. https://www.kaggle.com/gpreda/credit-card-fraud-detection-predictive-models