Title: Assignment 2: Credit Card Fraud Detection

Author: "put your student id here"

Introduction:

In assignment 2, you will get familier with Jupyter Notebook and Python for predictive data analytics task. Jupyter notebook is very similar to Rmarkdown,it is Python veriosn of Rmarkdown

To start with, you should install Jupyter Notenook on your laptop/PC. The easist way for beginners is to install Jupyter Notebook via Anaconda https://www.anaconda.com/distribution/. You can also install Jupyter Notebook from Python. A nice tutorial on Jupyter Notebook installation and usage: https://www.dataquest.io/blog/jupyter-notebook-tutorial/. Introduction to the history of Jupyter Notebook: https://www.youtube.com/watch?v=ctOM-Gza04Y

Your goal in this assignment is to compare multiple machine learning classifiers, namely Logistic Regression (covered in topic 1), Random Forest (typical bagging method), XGBoost on a popular predictive data analystics task, i.e., Credit Card Fraud Detection. You will also be asked to perform hyperparameter tuning for three classifiers using random search and grid search.

You can always add more code/markdown cells.

Dataset

The dataset we are using is from Kaggle: https://www.kaggle.com/mlg-ulb/creditcardfraud#creditcard.csv. 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.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we 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-dependant cost-senstive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

In [3]:
```
# import needed Python libraries, e.g., sklearn and xgboost
```

Task 1: load dataset and set up pipeline (10 points).

In reality, we usually use 10-fold cross validation rather than reporting evalution performance on one single split of data. Cross-validation: Ref. https://towardsdatascience.com/5-reasons-why-you-should-use-cross-validation-in-your-data-science-project-8163311a1e79

Using pandas to load and explore data, e.g., dataset = pd.read_csv(r"D:/Datasets/creditfraud.csv", sep=';'). Ref: https://www.datacamp.com/community/tutorials/pandas-read-csv

pandas is similar to dataframe in R. You can also use dataset.head(3) to check the initial 3 rows of your dataset.

Using scikit-learn package, you can easily build data analytics pipeline. e.g.,

```
from sklearn.model_selection import cross_val_score # for cross validation
from sklearn import datasets  # import build-in dataset
from sklearn import svm  # import model

iris = datasets.load_iris()  # load dataset
clf_svc_cv = svm.SVC(kernel='linear',C=1)  # build a support vector machine with parameters
scores_clf_svc_cv = cross_val_score(clf_svc_cv,iris.data,iris.target,cv=10)  # 10-fold cross validation
print(scores_clf_svc_cv) # print results
print("Accuracy: %0.2f (+/- %0.2f)" % (scores_clf_svc_cv.mean(), scores_clf_svc_cv.std() * 2))  # print accuracy
```

https://scikit-learn.org/stable/tutorial/statistical_inference/model_selection.html

You can also use sklearn.model_selection.StratifiedKFold, so that you could iterate 10 folds and perform data preprocessing. e.g.,

```
from sklearn.model_selection import StratifiedKFold
from imblearn.over_sampling import SMOTE
cv = StratifiedKFold(n_splits=10)

for train_idx, test_idx, in cv.split(X, y):
   sm = SMOTE()
   X_train_oversampled, y_train_oversampled = sm.fit_sample(X_train, y_train)
   model = ...  # Choose a model here
   model.fit(X_train_oversampled, y_train_oversampled )
   y_pred = model.predict(X_test)
   print(f'For fold {fold}:')
   print(f'Accuracy: {model.score(X_test, y_test)}')
   print(f'f-score: {f1_score(y_test, y_pred)}')
```

In [ ]:
```

```

Task 2: model building and evaluation (40 points).

Create three models: Logistic Regression, Random Forest, Gradient Boost (XGBoost) using their default hyperparameter values. Report the default parameter values and model performance in terms of precision, recall, F-measure, and AUC. Note that you should perform evalution under the cross validation framework.

Note that Fraud is your positive class. You don't need to consider all hyperparameters provided by the tool, only focusing on three or four important ones.

We will apply SMOTE technique to deal with the highly imbalanced dataset. Ref. https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/ E.g.,

```
sm = SMOTE()
X_train_oversampled, y_train_oversampled = sm.fit_sample(X_train, y_train)
```

How to build random forest and xgboost model? E.g.,

```
from sklearn.ensemble import RandomForestClassifier
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, Y_train)
Y_pred = random_forest.predict(X_test)

import xgboost as xgb
gbm = xgb.XGBClassifier(max_depth=3, n_estimators=300, learning_rate=0.05).fit(X_train, Y_train)
Y_pred = gbm.predict(X_test)
```

In [ ]:
```

```

Answer: Please fill the following table based on your experiment results:

Classifier	Default Hyperparameter values	Precision	Recall	AUC
---	---	---	---	
---	---	---	---	

Task 3: hyperparameter tuning (40 points).

For each classifier, turn the hyperparameter using random search and grid search. Report the best performance for each classifier and their running time. Ref. https://scikit-learn.org/stable/auto_examples/model_selection/plot_randomized_search.html

In [ ]:
```

```

Answer: Please list your best performance model, hyper parameter values for the best model and running time below:

Task 4: take-away message (10 points)

Please summary your findings from the above analysis (at least 2 findings):