Prepare python environment

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

%matplotlib inline


random_state=5 # use this to control randomness across runs e.g., dataset partitior
```

Preparing the Credit Card Fraud Detection dataset (2 points)

The dataset 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. See here for details of the dataset. We will post process the data to balance both the classes indicating whether the transaction is fraud or not.

[ ] ↳ *19 cells hidden*

Training error-based models (18 points)

We will use the `sklearn` library to train a Multinomial Logistic Regression classifier and Support Vector Machines.

Exercise 1: Learning a Multinomial Logistic Regression classifier (4 points)

Use `sklearn`'s `SGDClassifier` to train a multinomial logistic regression classifier (i.e., using a one-versus-rest scheme) with Stochastic Gradient Descent. Review ch.7 and see here for more details.

Set the `random_state` as defined above, increase the `n_iter_no_change` to 1000 and `max_iter` to 10000 to facilitate better convergence.

Report the model's accuracy over the training and test sets.

```
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import accuracy_score


# Create Logistic Regression based classifier
clf = SGDClassifier(loss='log', random_state=random_state, n_iter_no_change=1000, m

# Train Classifer on training set
clf = clf.fit(X_train,y_train)

#Predict the response for train dataset
y_pred_train = clf.predict(X_train)

#Predict the response for test dataset
y_pred_test = clf.predict(X_test)

print("train accuracy: %.2f" % accuracy_score(y_train, y_pred_train))
print("test accuracy: %.2f" % accuracy_score(y_test, y_pred_test))
```

```
train accuracy: 0.95
test accuracy: 0.93
```

Explain any performance difference observed between the training and test datasets.

The classifier is slightly overfitting to the training dataset, resulting in lower accuracy on the test dataset.

Exercise 2: Learning a Support Vector Machine (SVM) (14 points)

Use `sklearn`'s `SVC` class to train an SVM (i.e., using a one-versus-one scheme). Review ch.7 and see here for more details.

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

Exercise 2a: Warm up (2 points)

Train an SVM with a linear kernel. Set the random_state to the value defined above. Keep all other parameters at their defaults.

Report the model's accuracy over the training and test sets.

```
clf = SVC(kernel='linear', random_state=random_state)
clf.fit(X_train,y_train)

print("training acc: %.2f" % accuracy_score(y_train, clf.predict(X_train)))
print("test acc: %.2f" % accuracy_score(y_test, clf.predict(X_test)))
```

```
training acc: 0.95
test acc: 0.94
```

Exercise 2b: Evaluate a polynomial kernel function (4 points)

Try fitting an SVM with a polynomial kernel function and vary the degree among {1, 2, 3, 4}. Note that degree=1 yields a linear kernel.

For each fitted classifier, report its accuracy over the training and test sets.

As before, set the random_state to the value defined above. Set the regularization strength `C=100`. When the data is not linearly separable, this encourages the model to fit the training data. Keep all other parameters at their default values.

```
for degree in [1,2,3,4]:
    clf = SVC(kernel='poly', random_state=random_state, degree=degree, C=100)
    clf.fit(X_train,y_train)

    print("Poly kernel, degree: %d" %degree)
    print("training acc: %.2f" % accuracy_score(y_train, clf.predict(X_train)))
    print("test acc: %.2f" % accuracy_score(y_test, clf.predict(X_test)))
```

```
Poly kernel, degree: 1
training acc: 0.95
test acc: 0.93
Poly kernel, degree: 2
training acc: 0.97
test acc: 0.86
Poly kernel, degree: 3
training acc: 0.99
test acc: 0.92
Poly kernel, degree: 4
training acc: 0.99
test acc: 0.89
```

▾ Explain the effect of increasing the degree of the polynomial.

Increasing the degree of the polynomial kernel function allows the model to better fit the training dataset. Consequently the model overfits to the training data and reduces generalization.

▾ Exercise 2c: Evaluate the radial basis kernel function (6 points)

Try fitting an SVM with a radial basis kernel function and vary the length-scale parameter given by γ among {0.1, 0.01,1,10, 100}.

For each fitted classifier, report its accuracy over the training and test sets.

As before, set the random_state to the value defined above. Set the regularization strength `C=100`. When the data is not linearly separable, this encourages the model to fit the training data (read more here). Keep all other parameters at their default values.

```
for gamma in [0.01,0.1,1,10,100]:
    clf = SVC(kernel='rbf', random_state=random_state, gamma=gamma, C=100)
    clf.fit(X_train,y_train)

    print("RBF kernel, gamma: ", gamma)
    print("training acc: %.2f" % accuracy_score(y_train, clf.predict(X_train)))
    print("test acc: %.2f" % accuracy_score(y_test, clf.predict(X_test)))
```

```
RBF kernel, gamma:  0.01
training acc: 0.97
test acc: 0.93
RBF kernel, gamma:  0.1
training acc: 1.00
test acc: 0.92
RBF kernel, gamma:  1
training acc: 1.00
test acc: 0.80
RBF kernel, gamma:  10
training acc: 1.00
test acc: 0.61
RBF kernel, gamma:  100
training acc: 1.00
test acc: 0.56
```

Comment on the effect of increasing/reducing the length-scale parameter γ. Also, compare the performance of the classifiers trained with RBF kernel function against those trained with the polynomial and linear kernel functions (i.e., Ex. 2b).

Increasing γ degrades generalization. γ determines when points are deemed close and far. Larger γ reduces the radius of influence of select training examples (support vectors), so that data points are only affected by a few nearby support vectors during inference. Smaller γ increases the radius of influence of select training examples (support vectors), so that data points are also affected by more distant support vectors during inference.

Increasing γ to values greater than 0.1 reduces generalization performance because the classifier effectively relies on a few training examples (support vectors) that are close to the data point in order to make its prediction. Learn more here.

Classifiers trained with the RBF kernel function overfit the training data more than those trained with the polynomial and linear kernel functions. This is because the RBF kernel function maps into an infinite-dimensional feature space. The increased overfitting, esp. at large γ, degrades generalization performance, resulting in less performant classifiers compared to classifiers trained with linear and polynomial kernel functions.

Exercise 2d: Briefly state the main difference between the logistic regression classifier and the SVM. (2 points)

SVMs are explicitly trained to learn decision boundaries with large margins separating the classes.