*You are currently looking at **version 1.2** of this notebook. To download notebooks and datafiles, as well as get help on Jupyter notebooks in the Coursera platform, visit the Jupyter Notebook FAQ course resource.*

Assignment 3 - Evaluation

In this assignment you will train several models and evaluate how effectively they predict instances of fraud using data based on this dataset from Kaggle.

Each row in `fraud_data.csv` corresponds to a credit card transaction. Features include confidential variables `V1` through `V28` as well as `Amount` which is the amount of the transaction.

The target is stored in the `class` column, where a value of 1 corresponds to an instance of fraud and 0 corresponds to an instance of not fraud.

In [1]:

```
import numpy as np
import pandas as pd
```

Question 1

Import the data from `fraud_data.csv`. What percentage of the observations in the dataset are instances of fraud?

This function should return a float between 0 and 1.

In [2]:

```
def answer_one():
    df = pd.read_csv('fraud_data.csv')
    y = df.iloc[:,-1]

    percent_fraud=len(y[y==1])/len(y)

    return (percent_fraud)
#answer_one()
#Function answer_one was answered incorrectly, 0.165 points were not awarded.
```

In [3]:

```
# Use X_train, X_test, y_train, y_test for all of the following questions
from sklearn.model_selection import train_test_split

df = pd.read_csv('fraud_data.csv')

X = df.iloc[:,:-1]
y = df.iloc[:,-1]

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

Question 2

Using `X_train`, `X_test`, `y_train`, and `y_test` (as defined above), train a dummy classifier that classifies everything as the majority class of the training data. What is the accuracy of this classifier? What is the recall?

This function should a return a tuple with two floats, i.e. `(accuracy score, recall score)`.

In [4]:

```
def answer_two():
    from sklearn.dummy import DummyClassifier
```

```
    from sklearn.metrics import accuracy_score, recall_score
    dummy_majority = DummyClassifier(strategy = 'most_frequent').fit(X_train, y_train)
    y_dummy_predictions = dummy_majority.predict(X_test)
    accuracy_score=accuracy_score(y_test, y_dummy_predictions)
    recall_score=recall_score(y_test, y_dummy_predictions)
    return (accuracy_score, recall_score)
#answer_two()
```

Question 3

Using X_train, X_test, y_train, y_test (as defined above), train a SVC classifer using the default parameters. What is the accuracy, recall, and precision of this classifier?

This function should a return a tuple with three floats, i.e. `(accuracy score, recall score, precision score)`.

In [5]:

```
def answer_three():
    from sklearn.metrics import accuracy_score, recall_score, precision_score
    from sklearn.svm import SVC
    svm = SVC().fit(X_train, y_train)
    svm_predicted = svm.predict(X_test)
    accuracy_score=accuracy_score(y_test, svm_predicted)
    recall_score=recall_score(y_test, svm_predicted)
    precision_score=precision_score(y_test, svm_predicted)
    return (accuracy_score, recall_score, precision_score)
#answer_three()
```

Question 4

Using the SVC classifier with parameters `{'C': 1e9, 'gamma': 1e-07}`, what is the confusion matrix when using a threshold of -220 on the decision function. Use X_test and y_test.

This function should return a confusion matrix, a 2x2 numpy array with 4 integers.

In [6]:

```
def answer_four():
    from sklearn.metrics import confusion_matrix
    from sklearn.svm import SVC
    svm = SVC(C=1e9, gamma=1e-07)
    y_scores_lr = svm.fit(X_train, y_train).decision_function(X_test)
    y_score_list = list(zip(y_test, y_scores_lr))
    y_predict_threshold=[]
    #for loop where every time the decision function "score" is greater than -220 make it a 1, els
e make it a 0
    for i in y_score_list:
        if i[1]>-220:
            y_predict_threshold.append(1)
        else:
            y_predict_threshold.append(0)
    confusion = confusion_matrix(y_test, y_predict_threshold)
    return confusion
#answer_four()
```

Question 5

Train a logisitic regression classifier with default parameters using X_train and y_train.

For the logisitic regression classifier, create a precision recall curve and a roc curve using y_test and the probability estimates for X_test (probability it is fraud).

Looking at the precision recall curve, what is the recall when the precision is `0.75`?

Looking at the roc curve, what is the true positive rate when the false positive rate is `0.16`?

This function should return a tuple with two floats, i.e. `(recall, true positive rate)`.

In [7]:

In [7]:

```
def answer_five():
    from sklearn.linear_model import LogisticRegression
    #from sklearn.metrics import precision_recall_curve
    from sklearn.metrics import roc_curve, auc
    import matplotlib.pyplot as plt
    lr = LogisticRegression()
    #use decision_function, not predict
    y_score_lr = lr.fit(X_train, y_train).decision_function(X_test)
    #fpr_lr, tpr_lr, _ = roc_curve(y_test, y_score_lr)
    #roc_auc_lr = auc(fpr_lr, tpr_lr)

    #after having done the plot and eye-balled it
    recall=0.8
    #after having done the plot and eye-balled it
    true_pos=0.9
    return (recall, true_pos)

#answer_five()
```

Question 6

Perform a grid search over the parameters listed below for a Logisitic Regression classifier, using recall for scoring and the default 3-fold cross validation.

`'penalty': ['l1', 'l2']`

`'C':[0.01, 0.1, 1, 10, 100]`

From `.cv_results_`, create an array of the mean test scores of each parameter combination. i.e.

	l1	l2
0.01	?	?
0.1	?	?
1	?	?
10	?	?
100	?	?

This function should return a 5 by 2 numpy array with 10 floats.

Note: do not return a DataFrame, just the values denoted by '?' above in a numpy array. You might need to reshape your raw result to meet the format we are looking for.

In [12]:

```
def answer_six():
    from sklearn.model_selection import GridSearchCV
    from sklearn.linear_model import LogisticRegression
    grid_values = {'C':[0.01, 0.1, 1, 10, 100], 'penalty': ['l1', 'l2']}
    lr=LogisticRegression()
    grid_lr_recall = GridSearchCV(lr, param_grid = grid_values, scoring = 'recall')
    grid_lr_recall.fit(X_train, y_train)
    #.cv_results_ returns a dictionary from 'mean_test_score' key get values
    #for l1 start at index 0 and get every other one (that's how they are arranged according to .c
v_results_['param_penalty'])
    l1=grid_lr_recall.cv_results_['mean_test_score'][0::2]
    #reshape it into the preferred shape
    l1=np.array(l1).reshape(5,1)
    #for l2 start at index 1 and get every other one
    l2=grid_lr_recall.cv_results_['mean_test_score'][1::2]
    #reshape it into the preferred shape
    l2=np.array(l2).reshape(5,1)
    return np.hstack((l1, l2))
#answer_six()
#Function answer_five was answered incorrectly, 0.167 points were not awarded. The code for this p
roblem is not compiling.
```

In [10]:

```
# Use the following function to help visualize results from the grid search
def GridSearch_Heatmap(scores):
    %matplotlib notebook
    import seaborn as sns
    import matplotlib.pyplot as plt
    plt.figure()
    sns.heatmap(scores.reshape(5,2), xticklabels=['l1','l2'], yticklabels=[0.01, 0.1, 1, 10, 100])
    plt.yticks(rotation=0);

#GridSearch_Heatmap(answer_six())
```