*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 (https://www.coursera.org/learn/python-machine-learning/resources/bANLa) 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 (https://www.kaggle.com/dalpozz/creditcardfraud).

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]:

```
%load_ext autoreload
%autoreload 2

import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.dummy import DummyClassifier
from sklearn.metrics import recall_score, precision_score, accuracy_score
from sklearn.metrics import confusion_matrix, precision_recall_curve, roc_curve, auc
from sklearn.svm import SVC

# Hide warnings
import warnings
warnings.filterwarnings('ignore')

# The following lines adjust the granularity of reporting
pd.options.display.max_rows = 10
pd.options.display.float_format = '{:.2f}'.format
```

In [2]:

```
# Loading the data
data = pd.read_csv('fraud_data.csv')
data.head()
```

Out[2]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	1.18	0.32	0.54	1.05	-0.37	-0.73	0.08	-0.07	-0.27	0.16	...	-0.11	-0.34	0.06	0.50	0.42	-0.58	0.02	0.02	4.67	0
1	0.68	-3.93	-3.80	-1.15	-0.74	-0.50	1.04	-0.63	-2.27	1.53	...	0.65	0.27	-0.98	0.17	0.36	0.20	-0.26	0.06	912.00	0
2	1.14	0.45	0.25	2.38	0.34	0.43	0.09	0.17	-0.81	0.78	...	-0.00	0.06	-0.12	-0.30	0.65	0.12	-0.01	-0.01	1.00	0
3	-1.11	-3.30	-0.18	-1.80	2.14	-1.68	-2.02	-0.01	-0.17	0.87	...	0.13	0.33	0.93	-0.05	-1.89	-0.58	0.27	0.41	62.10	0
4	-0.31	0.87	-0.12	-0.63	2.65	3.43	0.19	0.67	-0.44	0.13	...	-0.31	-0.80	-0.06	0.95	-0.43	0.16	0.08	-0.02	2.67	0

5 rows × 30 columns

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 [3]:

```
#  Imbalanced Classification
#  Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

data['Class'].value_counts()
```

Out[3]:

```
0    21337
1      356
Name: Class, dtype: int64
```

In [4]:

```
def answer_one():

    # Your code here
    fraud = len(data[data['Class'] == 1]) / len(data)

    return fraud

answer_one()
```

Out[4]:

```
0.016410823768035772
```

In [5]:

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

X = data.iloc[:,:-1]
y = data.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 [6]:

```
def answer_two():
#     from sklearn.dummy import DummyClassifier
#     from sklearn.metrics import recall_score, accuracy_score

    # Your code here
    dummy_majority = DummyClassifier(strategy='most_frequent').fit(X_train,y_train)
    y_majority_predicted = dummy_majority.predict(X_test)

    accuracy = accuracy_score(y_test, y_majority_predicted )
    recall = recall_score(y_test, y_majority_predicted)

    return accuracy, recall

answer_two()
```

Out[6]:

```
(0.98525073746312686, 0.0)
```

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 [7]:

```
def answer_three():
#     from sklearn.metrics import recall_score, precision_score
#     from sklearn.svm import SVC

    # Your code here
    model = SVC(kernel='rbf', C=1).fit(X_train,y_train)
    y_predicted = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_predicted)
    recall = recall_score(y_test, y_predicted)
    precision = precision_score(y_test, y_predicted)

    return accuracy, recall, precision

answer_three()
```

Out[7]:

```
(0.99078171091445433, 0.375, 1.0)
```

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 [8]: def answer_four():
        #     from sklearn.metrics import confusion_matrix
        #     from sklearn.svm import SVC

            # Your code here
            model = SVC(kernel='rbf', C=1e9, gamma=1e-07).fit(X_train,y_train)
            y_predicted = model.decision_function(X_test) > -220
            cm = confusion_matrix(y_test, y_predicted)

            return cm

        answer_four()
```

```
Out[8]: array([[5320,   24],
               [  14,   66]])
```

Question 5

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

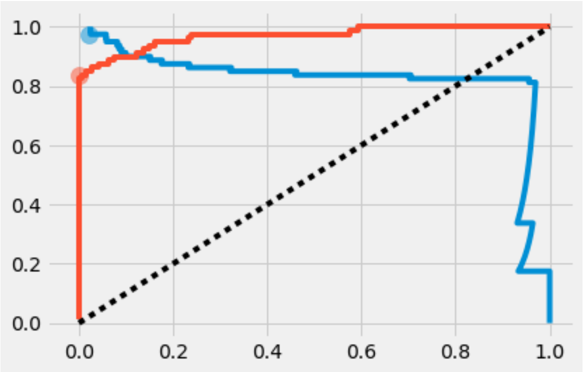
For the logisitic regression classifier, create a precision recall curve and a roc curve using y_test and the probability estimates for X_test (probability it is fraud).

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 [9]: def create_plot():

            %matplotlib inline
            import matplotlib.pyplot as plt
            import matplotlib.style as style
            style.use('fivethirtyeight')

            model = LogisticRegression()
            model_scores = model.fit(X_train,y_train).decision_function(X_test)

            precision, recall, thresholds = precision_recall_curve(y_test, model_scores)
            fpr, tpr, _ = roc_curve(y_test, model_scores)
            roc_auc = auc(fpr, tpr)

            plt.figure()
            plt.plot(precision,recall)
            plt.plot(fpr,tpr)
            plt.plot([0, 1], [0, 1], color='black', linestyle=':')
            plt.scatter(0.02, 0.975, marker='o', s=150, alpha=0.5)
            plt.scatter(0, 0.835, marker='o', s=150, alpha=0.5)
            plt.show()

        create_plot();
```



In [10]:

```
def answer_five():

    # Your code here
    model = LogisticRegression()
    model_scores = model.fit(X_train,y_train).decision_function(X_test)

    precision, recall, thresholds = precision_recall_curve(y_test, model_scores)
    fpr, tpr, _ = roc_curve(y_test, model_scores)
    roc_auc = auc(fpr, tpr)

    return 0.835, 0.975

answer_five()
```

Out[10]:

```
(0.835, 0.975)
```

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 [11]:

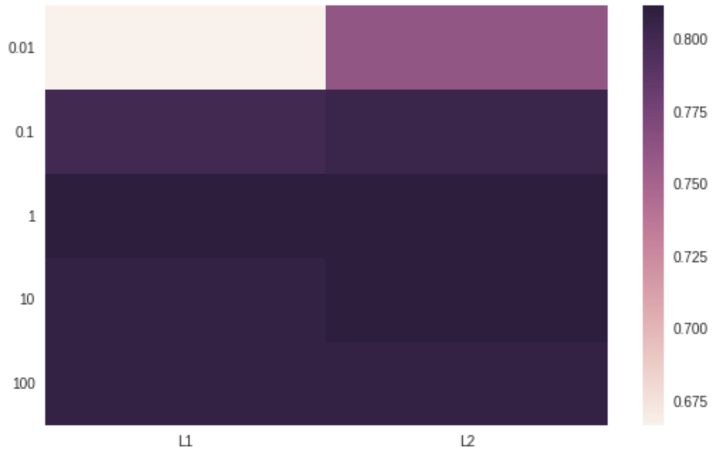
```
def answer_six():
#     from sklearn.model_selection import GridSearchCV
#     from sklearn.linear_model import LogisticRegression

    # Your code here
    model = LogisticRegression().fit(X_train, y_train)
    parameters ={'penalty': ['l1', 'l2'], 'C':[0.01, 0.1, 1, 10, 100]}

    grid = GridSearchCV(model, param_grid = parameters, scoring = 'recall',cv=3).fit(X_train, y_train)
    cv_results = grid.cv_results_['mean_test_score'].reshape(5,2)

    return cv_results

answer_six()
```

Out[11]:

```
array([[ 0.66666667,  0.76086957],
       [ 0.80072464,  0.80434783],
       [ 0.8115942 ,  0.8115942 ],
       [ 0.80797101,  0.8115942 ],
       [ 0.80797101,  0.80797101]])
```

In [12]:

```
# Use the following function to help visualize results from the grid search
def GridSearch_Heatmap(scores):
    %matplotlib inline
    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())
```


0.01
0.1
1
10
100
L1
L2
0.800
0.775
0.750
0.725
0.700
0.675