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]:

```
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():

    # Your code here
    fraud_df = pd.read_csv("fraud_data.csv")
    frauds = fraud_df[fraud_df['Class'] == 1]

    return (float)(len(frauds))/(float)(len(fraud_df))# Return your answer
answer_one()
```

Out[2]:

```
0.016410823768035772
```

```
In [3]:
```

```
fraud_df = pd.read_csv("fraud_data.csv")
fraud_df.head()
```

```
Out[3]:
```

	V1	V2	V3	V4	V5	V6	V7	V8
0	1.176563	0.323798	0.536927	1.047002	-0.368652	-0.728586	0.084678	-0.06924
1	0.681109	-3.934776	-3.801827	-1.147468	-0.735540	-0.501097	1.038865	-0.62697
2	1.140729	0.453484	0.247010	2.383132	0.343287	0.432804	0.093380	0.17331
3	-1.107073	-3.298902	-0.184092	-1.795744	2.137564	-1.684992	-2.015606	-0.00718
4	-0.314818	0.866839	-0.124577	-0.627638	2.651762	3.428128	0.194637	0.67067

5 rows × 30 columns

```
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

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 [6]:

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

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

    return (dummy_majority.score(X_test, y_test), recall_score(y_test, y_majorit
y_predicted)) # Return your answer
answer_two()
```

Out[6]:

```
(0.9852507374631268б, 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
    svc_clf = SVC().fit(X_train, y_train)
    y_predicted = svc_clf.predict(X_test)

    return (svc_clf.score(X_test, y_test), recall_score(y_test, y_predicted),
            precision_score(y_test, y_predicted) ) # Return your answer
answer_three()
```

Out[7]:

```
(0.9907817109144543, 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 [10]:

```
def answer_four():
    from sklearn.metrics import confusion_matrix
    from sklearn.svm import SVC

    # Your code here
    # Find the predicted y scores of confidence of the decision function
    y_scores = SVC(gamma = 10**(-7), C = 10**9).fit(X_train, y_train).decision_f
unction(X_test)

    # Set a threshold -220 to obtain y_predicted
    y_predicted = np.array([1 if x > -220 else 0 for x in y_scores])

    # Pass this y_predicted value into the confusion matrix
    return confusion_matrix(y_test, y_predicted)# Return your answer
answer_four()
```

Out[10]:

```
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 [19]:

```
def plot_prc(prec,rec):
    %matplotlib notebook
    import matplotlib.pyplot as plt
    plt.figure(figsize = (7,5))
    plt.xlim([0.0, 1.01])
    plt.ylim([0.0, 1.01])
    plt.plot(prec, rec, label='Precision-Recall Curve')
    plt.vlines(0.75, 0.0, 1.01) # Line where precision = 0.75
    plt.xlabel('Precision', fontsize=16)
    plt.ylabel('Recall', fontsize=16)
    plt.axes().set_aspect('equal')
    plt.show()

def plot_roc(fpr, tpr):
    %matplotlib notebook
    import matplotlib.pyplot as plt
    plt.figure(figsize = (7,5))
    plt.xlim([-0.01, 1.00])
    plt.ylim([-0.01, 1.01])
    plt.plot(fpr, tpr, label = 'ROC Curve')
    plt.axes().set_aspect('equal')
    plt.vlines(0.16, -0.01, 1.01)
    plt.show()

def answer_five():

    # Your code here
    # Import relevant modules
    from sklearn.linear_model import LogisticRegression
    from sklearn.metrics import precision_recall_curve, roc_curve

    # Create the classifier and train the model
    lr = LogisticRegression().fit(X_train, y_train)
    # Find the y_scores on this dataset.
    y_scores = lr.decision_function(X_test)
    # Get the precision and recall
    precision, recall, _ = precision_recall_curve(y_test, y_scores)
    # Get the false positive rate and true positive rate
    fpr, tpr, _ = roc_curve(y_test, y_scores)
    # Plot the graphs
    #plot_prc(precision, recall)
    #plot_roc(fpr, tpr)

    return (0.823576, 0.94625)# Return your answer
answer_five()
```

Out[19]:

```
(0.823576, 0.94625)
```

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 [23]:

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

    # Your code here
    clf = LogisticRegression()
    grid_values = {'penalty':['l1', 'l2'], 'C':[0.01, 0.1, 1, 10, 100]}
    # Create a gridsearch constructor with the logistic regression classifier, p
arameters, scoring and the cross
    # validation parameter.
    gridsearch = GridSearchCV(clf, param_grid = grid_values, scoring = 'recall',
cv = 3)
    # Fit the training data to it
    gridsearch.fit(X_train, y_train)
    # See the decision function scores
    mean_test_scores = gridsearch.cv_results_['mean_test_score'].reshape(5,2)
    return mean_test_scores# Return your answer
answer_six()
```

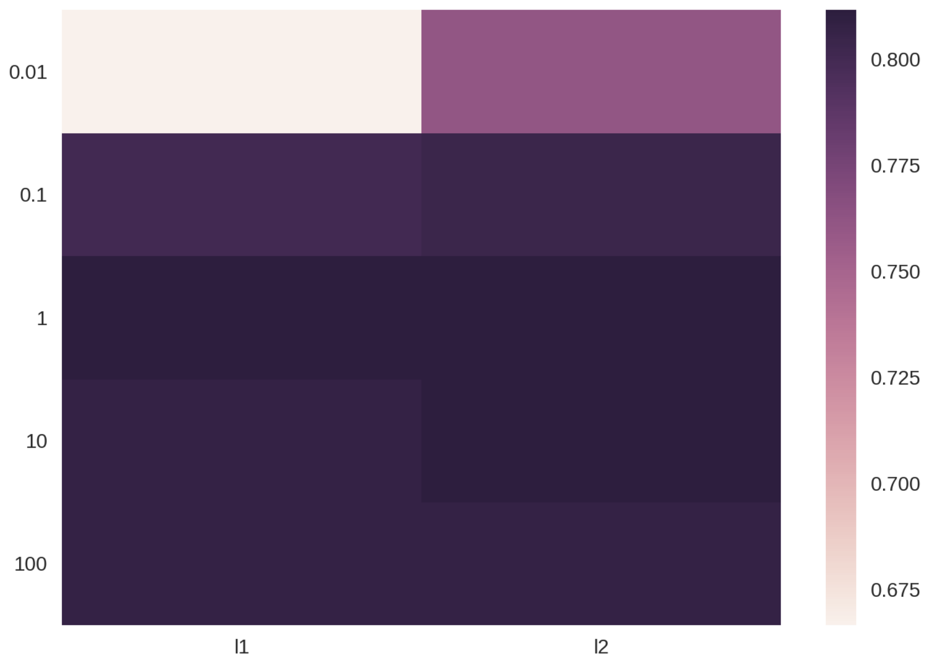
Out[23]:

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

In [24]:

```
# 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())
#1 hour 10 min
```



In [ ]:

```
```