Assignment 3

April 10, 2020

---

*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.*

---

1 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
```

1.0.1 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
            df = pd.read_csv('fraud_data.csv')
            total = df.Class.count()
            fraud = df[df.Class == 1]['Class'].count()
            percent = fraud / total

            return percent

        answer_one()

Out[2]: 0.016410823768035772
```

```
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)
```

1.0.2 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 recall_score,accuracy_score

             # Your code here
             dummy_majority = DummyClassifier(strategy = 'most_frequent').fit(X_trai
             # Therefore the dummy 'most_frequent' classifier always predicts class
             y_dummy_predictions = dummy_majority.predict(X_test)
             accuracy_score = dummy_majority.score(X_test, y_test)
             recall = recall_score(y_test.values,y_dummy_predictions)

             return  (accuracy_score, recall)
         answer_two()

Out[4]:  (0.98525073746312686, 0.0)
```

1.0.3 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 recall_score, precision_score
             from sklearn.svm import SVC

             # Your code here
             svm = SVC().fit(X_train, y_train)
             accuracy = svm.score(X_test, y_test)
             y_predicted = svm.predict(X_test)
             recall = recall_score(y_test.values,y_predicted)
```

```
        precision = precision_score(y_test.values,y_predicted)


            return (accuracy, recall, precision )
        answer_three()

Out[5]: (0.99078171091445433, 0.375, 1.0)
```

1.0.4 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 [11]: def answer_four():
            from sklearn.metrics import confusion_matrix
            from sklearn.svm import SVC

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

            return confusion
        answer_four()

Out[11]: array([[5320,    24],
               [  14,    66]])
```

1.0.5 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 [32]: def answer_five():

            # Your code here
            #% matplotlib notebook
            from sklearn.linear_model import LogisticRegression
            from sklearn.metrics import roc_curve, auc
            from sklearn.metrics import precision_recall_curve
            #import matplotlib.pyplot as plt

            lr = LogisticRegression().fit(X_train, y_train)
            y_scores_lr = lr.decision_function(X_test)
```

```
                precision, recall, thresholds = precision_recall_curve(y_test.values,
                y_score_lr = lr.fit(X_train, y_train).decision_function(X_test)
                fpr_lr, tpr_lr, _ = roc_curve(y_test.values, y_score_lr)
                roc_auc_lr = auc(fpr_lr, tpr_lr)
                #plt.figure()
                #plt.plot(precision, recall, label='Precision-Recall Curve')
                #plt.xlabel('Precision', fontsize=16)
                #plt.ylabel('Recall', fontsize=16)
                #plt.show()
                #plt.figure()
                #plt.plot(fpr_lr, tpr_lr, lw=3, label='LogRegr ROC curve (area = {:0.2
                #plt.xlabel('False Positive Rate', fontsize=16)
                #plt.ylabel('True Positive Rate', fontsize=16)
                #plt.title('ROC curve (1-of-10 digits classifier)', fontsize=16)
                #plt.legend(loc='lower right', fontsize=13)
                #plt.show()
                return (0.83, 0.94)
            answer_five()

<IPython.core.display.Javascript object>


<IPython.core.display.HTML object>


<IPython.core.display.Javascript object>


<IPython.core.display.HTML object>


Out[32]: (array([ 1.    ,  0.9875,  0.9875, ...,  0.025 ,  0.0125,  0.    ]),
          array([ 0.0125,  0.175 ,  0.175 ,  0.3375,  0.3375,  0.8125,  0.8125,
                  0.825 ,  0.825 ,  0.8375,  0.8375,  0.85  ,  0.85  ,  0.8625,
                  0.8625,  0.875 ,  0.875 ,  0.8875,  0.8875,  0.9   ,  0.9   ,
                  0.9125,  0.9125,  0.925 ,  0.925 ,  0.9375,  0.9375,  0.95  ,
                  0.95  ,  0.95  ,  0.95  ,  0.95  ,  0.95  ,  0.9625,  0.9625,
                  0.975 ,  0.975 ,  0.975 ,  0.975 ,  0.975 ,  0.975 ,  0.9875,
                  0.9875,  1.    ,  1.    ,  1.    ,  1.    ,  1.    ,  1.    ]))
```

1.0.6 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 [36]: def answer_six():
             from sklearn.model_selection import GridSearchCV
             from sklearn.linear_model import LogisticRegression

             # Your code here
             lr = LogisticRegression().fit(X_train, y_train)
             grid_values = {'penalty': ['l1', 'l2'], 'C': [0.01, 0.1, 1, 10, 100]}
             grid_clf_rec = GridSearchCV(lr, param_grid = grid_values, scoring = 'r
             grid_clf_rec.fit(X_train, y_train)
             return grid_clf_rec.cv_results_['mean_test_score'].reshape(5,2)




         answer_six()

Out[36]: array([[ 0.66666667,  0.76086957],
                [ 0.80072464,  0.80434783],
                [ 0.8115942 ,  0.8115942 ],
                [ 0.80797101,  0.8115942 ],
                [ 0.80797101,  0.80797101]])

In [ ]: # Use the following function to help visualize results from the grid searc
        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=[
            plt.yticks(rotation=0);

        #GridSearch_Heatmap(answer_six())
```