

Assignment 3

August 30, 2017

You are currently looking at **version 1.1** 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')
    Bin = np.bincount(df['Class'])
    fraud = Bin[1]/(Bin[0] + Bin[1])
    return fraud
```

```
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 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_train, y_train)
        y_dummy_predictions = dummy_majority.predict(X_test)
        dum_accuracy = accuracy_score(y_test, y_dummy_predictions)
        dum_recall = recall_score(y_test, y_dummy_predictions)
        return (dum_accuracy, dum_recall)
```

1.0.3 Question 3

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

This function should 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, accuracy_score
        from sklearn.svm import SVC

        # Your code here
        svm = SVC().fit(X_train, y_train)
        y_predicted = svm.predict(X_test)
        d_accuracy = accuracy_score(y_test, y_predicted)
        d_recall = recall_score(y_test, y_predicted)
        d_precision = precision_score(y_test, y_predicted)
        return (d_accuracy, d_recall, d_precision) # Return your answer
```

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 [6]: def answer_four():
        from sklearn.metrics import confusion_matrix
        from sklearn.svm import SVC
        threshold = -220
        # Your code here
        model = SVC(kernel='rbf', C=1e9, gamma = 1e-07)
        y_pred_sc = model.fit(X_train, y_train).decision_function(X_test)
        y_predict_threshold = y_pred_sc > threshold
        confusion = confusion_matrix(y_test, y_predict_threshold)
        return confusion # Return your answer
```

1.0.5 Question 5

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

For the logistic 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 [10]: def answer_five():

        # Your code here
        from sklearn.linear_model import LogisticRegression
        #import matplotlib.pyplot as plt
        lr = LogisticRegression()
        y_proba_lr = lr.fit(X_train, y_train).decision_function(X_test)
        #lr_prob = LogisticRegression().fit(X_train, y_train).predict_proba(X_test)
        #lr_predicted = lr.predict(X_test)

        from sklearn.metrics import precision_recall_curve
        precision, recall, thresholds = precision_recall_curve(y_test, y_proba_lr)
        #plt.show()

        from sklearn.metrics import roc_curve, auc
        fpr_lr, tpr_lr, _ = roc_curve(y_test, y_proba_lr)
        roc_auc_lr = auc(fpr_lr, tpr_lr)
        #plt.show()
        return (0.82, 0.95) # Return your answer
```

1.0.6 Question 6

Perform a grid search over the parameters listed below for a Logistic 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.

```
In [8]: def answer_six():  
        from sklearn.model_selection import GridSearchCV  
        from sklearn.linear_model import LogisticRegression  
  
        # Your code here  
        grid_values = {'penalty': ['l1', 'l2'], 'C':[0.01, 0.1, 1, 10, 100]}  
        grid_clf_rec = GridSearchCV(LogisticRegression(), \  
                                     param_grid = grid_values, cv=3, scoring = 'recall')  
        grid_clf_rec.fit(X_train, y_train)  
        #y_rec = grid_clf_rec.decision_function(X_test)  
        mean = grid_clf_rec.cv_results_['mean_test_score']  
        res = mean.reshape(5,2)  
        return res # Return your answer  
  
In [9]: # 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())
```

```
In [ ]:
```