

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](https://www.coursera.org/learn/python-machine-learning/resources/bANLa) (<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) (<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 [3]:

```
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 [5]:

```
def answer_one():
    # Your code here
    df = pd.read_csv('fraud_data.csv')
    ans = np.sum(df['Class'])/len(df.index)
    return ans # Return your answer
```

In [4]:

```
# 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 [8]:

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

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

    accuracy = accuracy_score(y_test, y_dummy_predictions)
    recall = recall_score(y_test, y_dummy_predictions)
    ans = (accuracy, recall)
    return ans # Return your answer
```

Out[8]:

(0.98525073746312686, 0.0)

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

In [11]:

```
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)
    svm_predicted = svm.predict(X_test)

    accuracy = svm.score(X_test, y_test)
    recall = recall_score(y_test, svm_predicted)
    precision = precision_score(y_test, svm_predicted)
    ans = (accuracy, recall, precision)
    return ans # Return your answer
```

Out[11]:

(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 [12]:

```
def answer_four():  
    from sklearn.metrics import confusion_matrix  
    from sklearn.svm import SVC  
  
    # Your code here  
    svm = SVC(C = 1e9, gamma = 1e-07).fit(X_train, y_train)  
    svm_predicted = svm.decision_function(X_test)  
    svm_predicted[svm_predicted > -220] = 1;  
    svm_predicted[svm_predicted <= -220] = 0;  
    confusion = confusion_matrix(y_test, svm_predicted)  
    return confusion # Return your answer
```

Out[12]:

```
array([[5320,  24],  
       [ 14,  66]])
```

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 [6]:

```
def answer_five():

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

    # lr = LogisticRegression().fit(X_train, y_train)
    # lr_predicted = lr.predict_proba(X_test)
    # precision, recall, thresholds = precision_recall_curve(y_test, lr_predicted[:,1])
    # plt.figure()
    # plt.xlim([0.0, 1.01])
    # plt.ylim([0.0, 1.01])
    # plt.plot(precision, recall, Label='Precision-Recall Curve')
    # plt.xlabel('Precision', fontsize=13)
    # plt.ylabel('Recall', fontsize=13)
    # plt.legend(loc="top right")
    # plt.axes().set_aspect('equal')
    # plt.show()

    # fpr_lr, tpr_lr, _ = roc_curve(y_test, lr_predicted[:,1])
    # roc_auc_lr = auc(fpr_lr, tpr_lr)
    # plt.figure()
    # plt.xlim([-0.01, 1.00])
    # plt.ylim([-0.01, 1.01])
    # plt.plot(fpr_lr, tpr_lr, lw=2)
    # plt.xlabel('False Positive Rate', fontsize=13)
    # plt.ylabel('True Positive Rate', fontsize=13)
    # plt.title('ROC curve', fontsize=13)
    # plt.plot([0.16, 0.16], [0, 1], color='navy', lw=2, linestyle='--')
    # plt.axes().set_aspect('equal')
    # plt.show()

    ans = (0.8, 0.95)
    return ans # Return your answer
answer_five()
```

Out[6]:

(0.8, 0.95)

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	?	?

	l1	l2
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 [7]:

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

    # Your code here
    penalty = ['l1', 'l2']
    C = [0.01, 0.1, 1, 10, 100]
    param_grid = {'penalty' : penalty, 'C': C}
    clf = LogisticRegression()
    grid_search = GridSearchCV(clf, param_grid, scoring='recall', cv=3)
    grid_search.fit(X_train, y_train)
    # grid_search.cv_results_
    ans = (grid_search.cv_results_['split0_test_score'] +
           grid_search.cv_results_['split1_test_score'] +
           grid_search.cv_results_['split2_test_score'])/3
    ans = ans.reshape(5,2)

    return ans # Return your answer
```

Out[7]:

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

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())
```