```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Nov 22 18:24:59 2021

@author: michaln
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

#%%

# Segment 1: show what a sigmoid is with some random data, explain
# how it classifies the data into either 0 or 1.

# The random data:
points = [
    [-5, -6, -7, 8, 10, 0],
    [0, 0, 0, 1, 1, 0]
    ]

# Parameters that determine the steepness (a) and offset (b) of the
# curve:
a = 1
b = 5

# Points for plotting the sigmoid:
x = np.linspace(-10, 10, 100)
z = 1 / (1+np.exp(- a*x + b))

plt.scatter(points[0], points[1], marker='x', s=250)
plt.grid()
plt.plot(x, z, color='red')
plt.show()

#%%

# Segment 2: introduce logistic regression with university admission
# data:

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

#%%

candidates = {
    'gpa' : [4., 3.9, 3.3, 3.7, 3.9, 3.7, 2.3, 3.3, 3.3, 1.7, 2.7,
3.7, 3.7, 3.3, 3.3, 3.0, 2.7, 3.7, 2.7, 2.3, 3.3, 2.0, 2.3, 2.7,
3.0, 3.3, 3.7, 2.3, 3.7, 3.3, 3.0, 2.7, 4.0, 3.3, 3.3, 2.3, 2.7,
3.3, 1.7, 3.7],
    'admitted' : [1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0,
1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0 , 0, 1, 1, 1, 0, 0, 0, 0,
```

```
1]
    }

# Convert to dataframe because dictionaries suck and dataframes are
awesome:
grade_df = pd.DataFrame(candidates)

plt.scatter(grade_df['gpa'], grade_df['admitted'])
plt.show()

#%%

# Use sklearn to split the data into an trainig and testing sample.
# Discuss briefly how this is especially importnt in machine
learning.
X_train, X_test, Y_train, Y_test = train_test_split(grade_df['gpa'],

grade_df['admitted'],
                                                    test_size=0.33,
                                                    random_state=0
                                                    )

# Logistic regression:
clf = LogisticRegression().fit(X_train.values.reshape(-1, 1),
Y_train)

#%%

# Create a prediction:
y_pred = clf.predict(X_test.values.reshape(-1, 1))

#plot together with real values
plt.scatter(X_test.values.reshape(-1, 1), Y_test.values.reshape(-1,
1), marker='+', c='navy', s=100, label='Real vals')
plt.scatter(X_test.values.reshape(-1, 1), y_pred, marker='x',
c='red', s=100, label='Prediction')
plt.grid(True)
plt.ylim([-0.1, 1.1])
plt.legend()
plt.show()

#%%

# Calculate the confusion matrix. Discuss what it is.
confusion_matrix = pd.crosstab(Y_test, y_pred)

# Use heatmaps from seaborn to visualize the confusion matrix:
import seaborn as sn

sn.heatmap(confusion_matrix, annot=True)

#%%

# Segment 3: use multivariable logistic regression to predict credit
```

```
score based
# one a huge number of parameters.

# The column names of the data
column_names = [
    'existingchecking', 'duration', 'credithistory', 'purpose',
'creditamount',
    'savings', 'employmentsince', 'installmentrate', 'statussex',
'otherdebtors',
    'residencessince', 'property', 'age', 'otherinstallmentplans',
'housing',
    'existingcredits', 'job', 'peopleliable', 'telephone',
'foreignworker',
    'classification'
    ]

# Import the credit score data:
df_raw = pd.read_csv('logisticdata.data', names=column_names,
delimiter=' ')
# Change the classification score so that it is 0 or 1, instead of 1
or 2:
df_raw['classification'].replace([1, 2], [1, 0], inplace=True)

# We want to change the string entires to numbers so that we can use
them
# for the regression. For that we use sklearn:
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict

d = defaultdict(LabelEncoder)

# Now we want to divide the columns into wheter they correspond to
sting entries
# or to numerical entries:
numvars = [
    'duration', 'creditamount', 'installmentrate',
'residencessince', 'age',
    'existingcredits', 'peopleliable', 'classification'
    ]

catvars = [
    'existingchecking', 'credithistory', 'purpose', 'savings',
'employmentsince',
    'statussex', 'otherdebtors', 'property',
'otherinstallmentplans', 'housing',
    'job', 'telephone', 'foreignworker'
    ]

# Change the string entries to numbers accoring to a mapping:
labeldata = df_raw[catvars].apply(lambda x:
d[x.name].fit_transform(x))

# Show quickly what the mapping did:
for x in range(len(catvars)):
```

```
    print(catvars[x], ': ', df_raw[catvars[x]].unique())
    print(catvars[x], ': ', labeldata[catvars[x]].unique())

# Select
dummyv = pd.get_dummies(df_raw[catvars])

# Create the clean data:
df_clean = pd.concat([df_raw[numvars], dummyv], axis=1)

X = df_clean.drop('classification', axis=1)
Y = df_clean['classification']

# Split into test and train
X_train, X_test, Y_train, Y_test = train_test_split(X,
                                                    Y,
                                                    test_size=0.2,
                                                    random_state=0
                                                    )

# Logistic regression:
clf = LogisticRegression().fit(X_train, Y_train)
# Create a prediction:
y_pred = clf.predict(X_test)

# Calculate the confusion matrix.
confusion_matrix = pd.crosstab(Y_test, y_pred)
sn.heatmap(confusion_matrix, annot=True)

from sklearn import metrics

print('Accuracy: ', metrics.accuracy_score(Y_test, y_pred))
```