Case Study 3 Homework: Exercises 1-5

Exercise 1

1/1 point (graded)
Our first step is to import the dataset.

Instructions
Read in the data as a pandas dataframe using `pd.read_csv`. The data can be found at this link from anywhere and at this link from within the courseware.

This code will get you started:

```
import pandas as pd

# write your code here!
```

Taking a look at the first 5 rows of the dataset, how many wines in those 5 rows are considered high quality?

Answer = [1]

Code = [
```
ata = pd.read_csv("https://courses.edx.org/asset-
v1:HarvardX+PH526x+2T2019+type@asset+block@wine.csv",
                    index_col=0)

data.head()
fixed acidity      volatile acidity   citric acid residual sugar     chlorides
free sulfur dioxide       total sulfur dioxide     density      pH       sulphates
alcohol       quality       color high_quality
0      7.4    0.70   0.00   1.9    0.076  11.0   34.0   0.99783.51    0.56   9.4    5
red    0
1      7.8    0.88   0.00   2.6    0.098  25.0   67.0   0.99683.20    0.68   9.8    5
red    0
2      7.8    0.76   0.04   2.3    0.092  15.0   54.0   0.99703.26    0.65   9.8    5
red    0
```

```
3       11.2   0.28   0.56   1.9    0.075  17.0   60.0   0.99803.16   0.58   9.8    6
red     1
4       7.4    0.70   0.00   1.9    0.076  11.0   34.0   0.99783.51   0.56   9.4    5
red     0
```

]

Exercise 2

1/1 point (graded)

Next, we will inspect the dataset and perform some mild data cleaning.

Instructions

In order to get all numeric data, we will change the `color` column to an `is_red` column.

- If `color == 'red'`, we will encode a `1` for `is_red`.
- If `color == 'white'`, we will encode a `0` for `is_red`.

Create this new column, `is_red`. Drop the `color` column as well as `quality` and `high_quality`. We will predict the quality of wine using the numeric data in a later exercise

Store this all numeric data in a pandas dataframe called `numeric_data`.

How many red wines are in the dataset?

Answer = [1599]

Code = [

```
data["is_red"] = (data["color"] == "red").astype(int)
numeric_data = data.drop("color", axis=1)
```

```
numeric_data.groupby('is_red').count()
Unnamed: 0   fixed acidity      volatile acidity   citric acid residual sugar
chlorides    free sulfur dioxide      total sulfur dioxide     density      pH
sulphates    alcohol      quality      high_quality
is_red
0      4898   4898   4898   4898   4898   4898   4898   4898   4898   4898   4898   4898
4898   4898
1      1599   1599   1599   1599   1599   1599   1599   1599   1599   1599   1599   1599
1599   1599
```

]

Exercise 3

0.0/1.0 point (graded)

We want to ensure that each variable contributes equally to the kNN classifier, so we will need to scale the data by subtracting the mean of each variable (column) and dividing each variable (column) by its standard deviation. Then, we will use principal components to take a linear snapshot of the data from several different angles, with each snapshot ordered by how well it aligns with variation in the data. In this exercise, we will scale the numeric data and extract the first two principal components.

Instructions

- Scale the data using the `sklearn.preprocessing` function `scale()` on `numeric_data`.
- Convert this to a `pandas` dataframe, and store it as `numeric_data`.
- Include the numeric variable names using the parameter `columns = numeric_data.columns`.

- Use the `sklearn.decomposition` module `PCA()` and store it as `pca`.
- Use the `fit_transform()` function to extract the first two principal components from the data, and store them as `principal_components`.

Note: You may get a `DataConversionWarning`, but you can safely ignore it.

Fill in this code as you work:

```
import sklearn.preprocessing
scaled_data = 
numeric_data = 
```

```
import sklearn.decomposition
pca = 
principal_components = 
```

What is the shape of the new dataset?

Enter the first number here.
Answer = [6497]
Enter the second number here.
Answer = [2]

Code= [
```
import sklearn.preprocessing
scaled_data = sklearn.preprocessing.scale(numeric_data)
```

```
numeric_data = pd.DataFrame(scaled_data, columns = 
numeric_data.columns)
```

```
import sklearn.decomposition
pca = sklearn.decomposition.PCA(n_components=2)
principal_components = pca.fit_transform(numeric_data)
```

```
principal_components.shape
(6497, 2)
```

]

Exercise 4

0/1 point (graded)

In Exercise 4, we will plot the first two principal components of the covariates in the dataset. The high and low quality wines will be colored using red and blue, respectively.

Instructions

The first two principal components can be accessed using `principal_components[:,0]` and `principal_components[:,1]`. Store these as `x` and `y` respectively, and make a scatter plot of these first two principal components.

Consider how well the two groups of wines are separated by the first two principal components.

Fill in your code where indicated to make the plot:

```
       import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from matplotlib.backends.backend_pdf import PdfPages
observation_colormap = ListedColormap(['red', 'blue'])
x = # Enter your code here!
y = # Enter your code here!

plt.title("Principal Components of Wine")
plt.scatter(x, y, alpha = 0.2,
    c = data['high_quality'], cmap =
observation_colormap, edgecolors = 'none')
plt.xlim(-8, 8); plt.ylim(-8, 8)
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
```

Could you easily draw a linear boundary between the high and low quality wines using the first two principal components?

Yes

No
Correct

Code = [

```
x = principal_components[:,0]
y = principal_components[:,1]
```

]

Exercise 5

1/1 point (graded)

In Exercise 5, we will create a function that calculates the accuracy between predictions and outcomes.

Instructions

- Create a function `accuracy(predictions, outcomes)` that takes two lists of the same size as arguments and returns a single number, which is the percentage of elements that are equal for the two lists.
- Use `accuracy` to compare the percentage of similar elements in the `x` and `y` numpy arrays defined below.
- Print your answer.

Here's the sample code to get you started:

```
        import numpy as np
np.random.seed(1) # do not change this!

x = np.random.randint(0, 2, 1000)
y = np.random.randint(0 ,2, 1000)

def accuracy(predictions, outcomes):
    # write your code here!
```

What is the accuracy of the `x` predictions on the "true" outcomes `y`?

Answer = [51.5]

Code = [
```
def accuracy(predictions, outcomes):
    return 100*np.mean(predictions == outcomes)

print(accuracy(x,y))
51.5
```

]

Exercise 6

1/1 point (graded)
The dataset remains stored as `data`. Because most wines in the dataset are classified as low quality, one very simple classification rule is to predict that all wines are of low quality. In this exercise, we determine the accuracy of this simple rule.

Instructions
Use `accuracy()` to calculate how many wines in the dataset are of low quality. Do this by using 0 as the first argument, and `data["high_quality"]` as the second argument.

Print your result.

What proportion of wines in the dataset are of low quality?

Answer = [36.69385870401724]

Code = [
```
 print(accuracy(0, data["high_quality"]))
36.69385870401724
```

]

Exercise 7

1/1 point (graded)

In Exercise 7, we will use the kNN classifier from `scikit-learn` to predict the quality of wines in our dataset.

Instructions

- Use `knn.predict(numeric_data)` to predict which wines are high and low quality and store the result as `library_predictions`.
- Use `accuracy` to find the accuracy of your predictions, using `library_predictions` as the first argument and `data["high_quality"]` as the second argument.
- Print your answer. Is this prediction better than the simple classifier in Exercise 6?

Here's the sample code to get you started:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(numeric_data, data['high_quality'])
# Enter your code here!
```

What is the accuracy of the KNN classifier? Please round your answer to the nearest integer.

Answer = [100]

Code = [
```
library_predictions = knn.predict(numeric_data)
print(accuracy(library_predictions, data[“high_quality"]))
```

```
99.96921656148992
```

]

Exercise 8

1/1 point (graded)

Unlike the `scikit-learn` function, our homemade kNN classifier does not take any shortcuts in calculating which neighbors are closest to each observation, so it is likely too slow to carry out on the whole dataset. In this exercise, we will select a subset of our data to use in our homemade kNN classifier.

Instructions

Fix the random generator using `random.seed(123)`, and select 10 rows from the dataset using `random.sample(range(n_rows), 10)`. Store this selection as `selection`.

Use this sample code to get started:

```
      n_rows = data.shape[0]
# Enter your code here
```

What is the 10th random row selected?

Answer = [4392]

Code = [

```
  random.seed(123)
selection = random.sample(range(n_rows), 10)
selection
[428, 2192, 714, 6299, 3336, 2183, 882, 312, 3105, 4392]    ]
```