Module 3-2 Exercises

The Iris data set is an extremely popular data set for learning machine learning techniqu
data set contains 3 classes that each correspond to a type of Iris plant. Each class has 5(
points and each data point consists of 4 features.

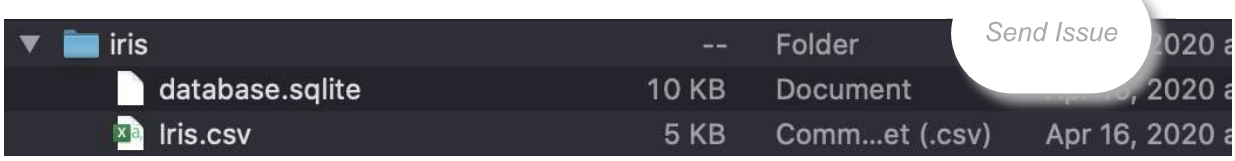
These features consist of the sepal length in cm, sepal width in cm, petal length in cm, a
width in cm.

Today we are going to use svm from the machine learning library, sklearn, in order to p
classes each Iris data point belongs to. Our goal is to pass the sepal length in cm, sepal
cm, petal length in cm, and petal width in cm in our svm model and have it predict one
classes of Iris flower this data could belong to.

Step 1

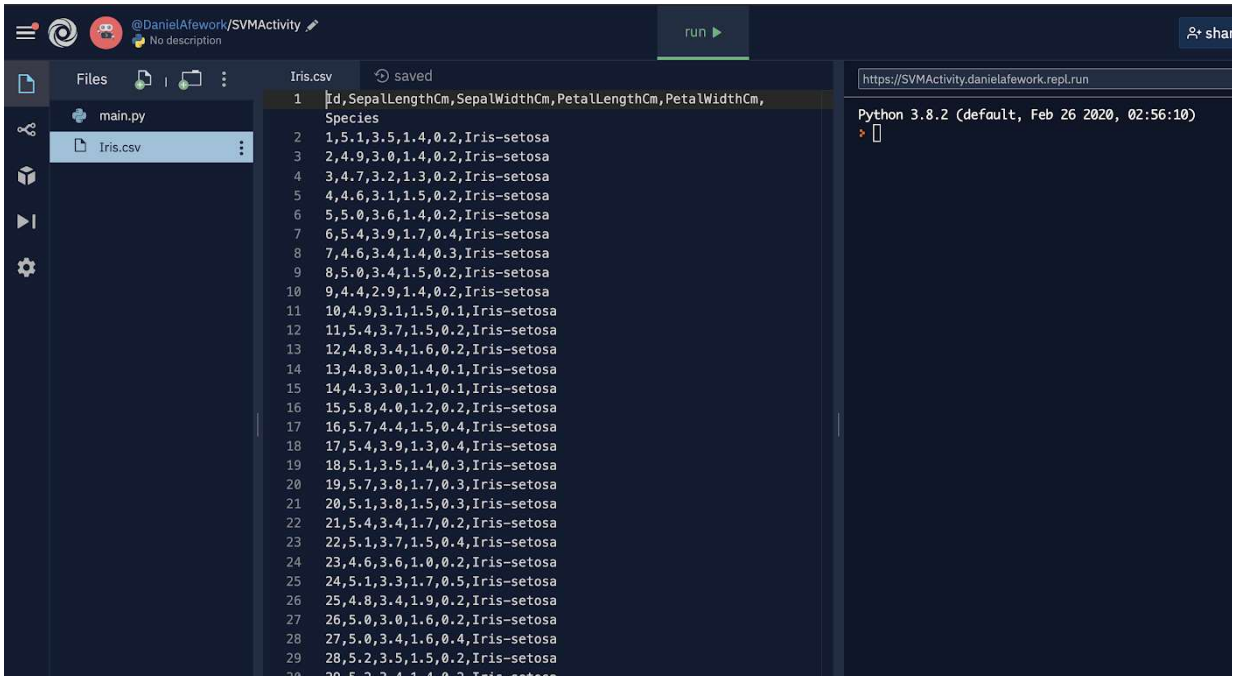
Before we can get started training our machine learning model we have to first get the
Download the Iris data set.

After clicking on the download button, a file called `archive.zip` should be downloaded
computer. Unzip the the file and to reveal a folder containing two files, `database.sqlit`
`Iris.csv` .



From here, drag and drop `Iris.csv` into a new repl.it. If you are using a local installatic
Python, drop it into whatever folder you will be making your program in. You should ha

title "Iris.csv" inside of your repl which has contents that look like the following.



Step 2

Let's store the dataset inside of a pandas dataframe so that we can use it to train our m first step is going to be to first import a few helpful libraries inside of the `main.py` file.

Begin by adding these lines to the top of `main.py` .

Python

```
from sklearn import svm

import pandas as pd
```

The first line will add the `svm` library from `sklearn` so that we can use the `svm` classifi model to train our data. The second line will add the `pandas` library so that we can use dataframes. `pandas` is a library used by data scientists that allows for easy data manipu

Now let's take the data from the `Iris.csv` file and add it to a `pandas` dataframe.

Python

```
df = pd.read_csv("Iris.csv")

print(df)
```

We can print the data frame by writing `print(df)` and get the following output. As you the information from the CSV file is now nicely organized inside of the `pandas` datafrar

```
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
0      1            5.1           3.5            1.4           0.2    Ir
1      2            4.9           3.0            1.4           0.2    Ir
2      3            4.7           3.2            1.3           0.2    Ir
3      4            4.6           3.1            1.5           0.2    Ir
4      5            5.0           3.6            1.4           0.2    Ir
..   ...            ...           ...            ...           ...
145  146            6.7           3.0            5.2           2.3  Iris-
146  147            6.3           2.5            5.0           1.9  Iris-
147  148            6.5           3.0            5.2           2.0  Iris-
148  149            6.2           3.4            5.4           2.3  Iris-
149  150            5.9           3.0            5.1           1.8  Iris-
```

Step 2

Let's take the "SepalLength", "SepalWidth", "PetalLength", and "PetalWidth" and separa into a smaller data frame so that we can use them to train our model.

Python

```
X = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWid
```

Now let's do the same thing, but for the species of flower.

Python

```
Y = df[['Species']]
```

Print both X and Y out to confirm that their contents are the separated flower data and data.

Python

```
print(X)

print(Y)
```

The output should look something like this.

```
     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
0              5.1           3.5            1.4           0.2
1              4.9           3.0            1.4           0.2
2              4.7           3.2            1.3           0.2
3              4.6           3.1            1.5           0.2
4              5.0           3.6            1.4           0.2
..             ...           ...            ...           ...
145            6.7           3.0            5.2           2.3
146            6.3           2.5            5.0           1.9
147            6.5           3.0            5.2           2.0
148            6.2           3.4            5.4           2.3
149            5.9           3.0            5.1           1.8

[150 rows x 4 columns]
            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 1 columns]
```

Step 3

Now, before we train our model, we have to encode the species data into numerical val species data is currently stored as a `string` value. However, the machine learning mod the data to be a numerical value.

To convert, we will use a `LabelEncoder`. Add this line of code to the top of your prograı

Python

```
from sklearn.preprocessing import LabelEncoder
```

Now we will encode the data using the `LabelEncoder`.

Python

```
le = LabelEncoder()

yEncoded = le.fit_transform(Y['Species'])
```

Note that we passed in `"Species"` instead of just `Y`. This is because the `le.fit_trans`
function is expecting a list and not the whole dataframe. `Y['Species']` will return the v
underneath the species column.

If we print `yEncoded`, the following values will be returned.

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

Step 4

The original text data for species has been encoded as numerical data! Now lets train o

Python

```
IrisPredictionModel = svm.SVC()

IrisPredictionModel.fit(X, yEncoded)
```

Now that our model is trained let's try to predict a flower.

Python

```
prediction = IrisPredictionModel.predict([ [4.9, 3, 1.4, .2]] )

print(prediction)
```

We are trying to predict a flower with `SepalLengthCm` of 4.9 , `SepalWidthCm` of 3 , `Petal` of 1.4 , and a `PetalWidthCm` of .2. After printing the prediction, we receive this value as

Step 5

Now, our final step will be to translate the encoded number back to its corresponding v

Python

```
returnToOrginal = le.inverse_transform(prediction)

print(returnToOrginal)
```

This is the final value that will be printed out after returning the encoded data back to it non numerical data.

Try predicting different numbers for the sepal length, width, and petal length and obser