

ASSIGNMENT 4 REGRESSION

NAHUSHA ACHARYA PES1201700044

KEERTHAN G PES1201700963

PUSHPENDAR SINGH PES1201700243

Regression is a statistical measurement used in finance, investing, and other disciplines that attempts to determine the strength of the relationship between one dependent variable (usually denoted by Y) and a series of other changing variables (known as independent variables).

Regression helps investment and financial managers to value assets and understand the relationships between variables, such as [commodity prices](#) and the stocks of businesses dealing in those commodities.

Applying regression on iris data set:

Code snippet:

```
data = pd.read_csv('C:/Users/Nahvsha acharya/Desktop/iris-flower-dataset/Iris.csv')
```

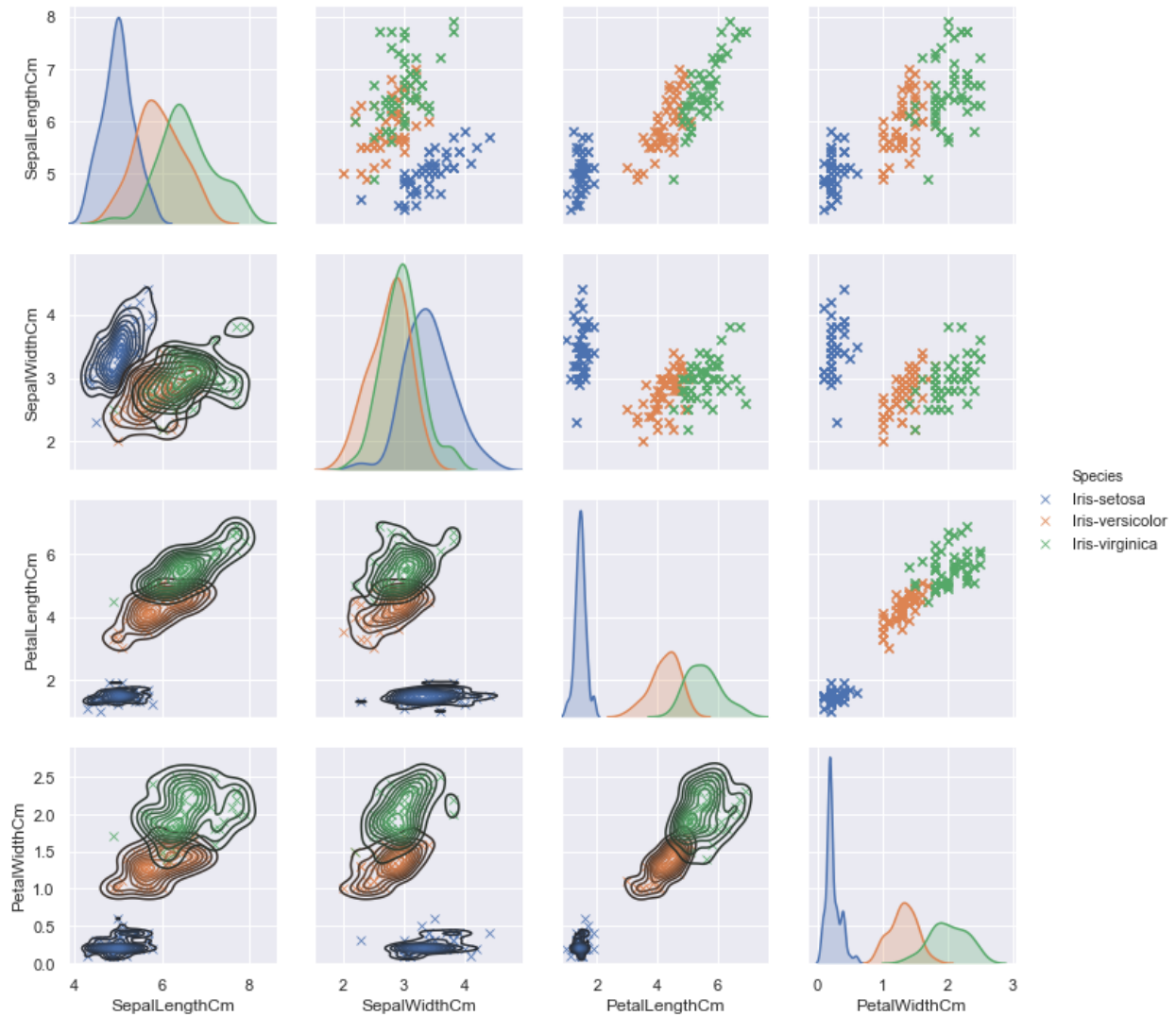
The screenshot shows a Jupyter Notebook titled 'REGRESSION' with a last checkpoint of 10/18/2019. The notebook is running on a local host (localhost:8888). The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running cells, and viewing output. The main area displays the Iris dataset data as a table with columns: Id, Sepal.LengthCm, Sepal.WidthCm, Petal.LengthCm, Petal.WidthCm, and Species. Below the data table, the output of the command `data.describe()` is shown, providing statistical summary information for the dataset. The output includes counts, mean, standard deviation, minimum, and maximum values for each column, as well as the 25th, 50th, and 75th percentiles.

	Id	Sepal.LengthCm	Sepal.WidthCm	Petal.LengthCm	Petal.WidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

```
In [7]: data.describe()
```

	Id	Sepal.LengthCm	Sepal.WidthCm	Petal.LengthCm	Petal.WidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

```
In [8]: rows, col = data.shape
print("Rows : %s, column : %s" % (rows, col))
```



```
sns.violinplot(x='SepalLengthCm', y='Species', data=data, inner='stick', palette='autumn')
```

```
plt.show()
```

```
sns.violinplot(x='SepalWidthCm', y='Species', data=data, inner='stick', palette='autumn')
```

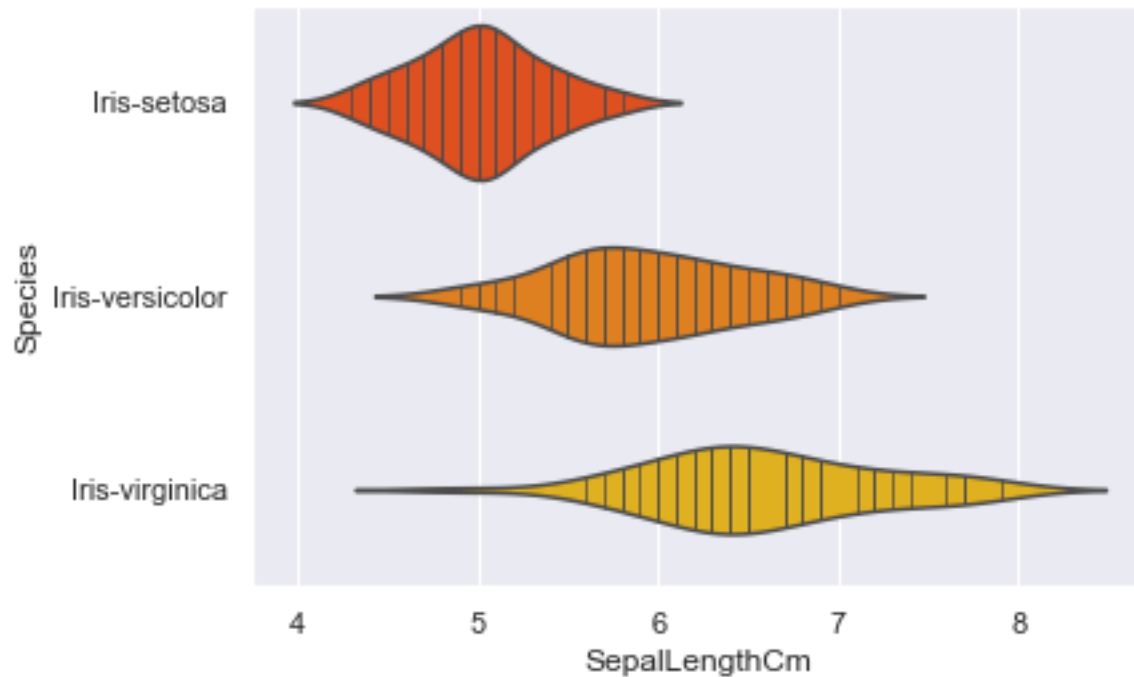
```
plt.show()
```

```
sns.violinplot(x='PetalLengthCm', y='Species', data=data, inner='stick', palette='autumn')
```

```
plt.show()
```

```
sns.violinplot(x='PetalWidthCm', y='Species', data=data, inner='stick', palette='autumn')
```

```
plt.show()
```



Let's train our model to compute values of theta

for i in range(iteration):

$J[i] = (1/(2 * rows) * np.sum((np.dot(X, theta.T) - y) ** 2))$

$theta -= ((learning_rate/rows) * np.dot((np.dot(X, theta.T) - y).reshape(1,rows), X))$

$prediction = np.round(np.dot(X, theta.T))$

$ax = plt.subplot(111)$

$ax.plot(np.arange(iteration), J)$

$ax.set_ylim([0,0.15])$

$plt.ylabel("Cost Values", color="Green")$

$plt.xlabel("No. of Iterations", color="Green")$

$plt.title("Mean Squared Error vs Iterations")$

$plt.show()$



Predictions

