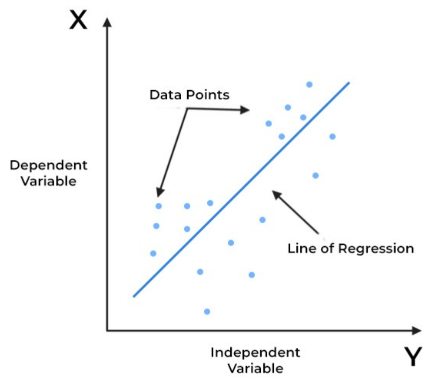
Machine Learning - CSE 6363-001 - Assignment 01
<u>Classification of Iris dataset using Linear Regression</u>

Name: Rajiv Nidumolu
UTA ID:1001866606

What is Regression : “*Regression is one of the modeling techniques in Machine Learning where we find or rather analyze how dependent are the predictor variable and dependent variable*” . We use different regression methods, to build machine learning models based on the analysis of the target and independent variables on the kind of linearity exhibited. Regression is primarily used in problems that deal with forecast trend, time series analysis.
We have six types of regression techniques :



- Linear Regression
- Logistic Regression
- Ridge Regression
- Lasso Regression
- Polynomial Regression
- Bayesian Linear Regression

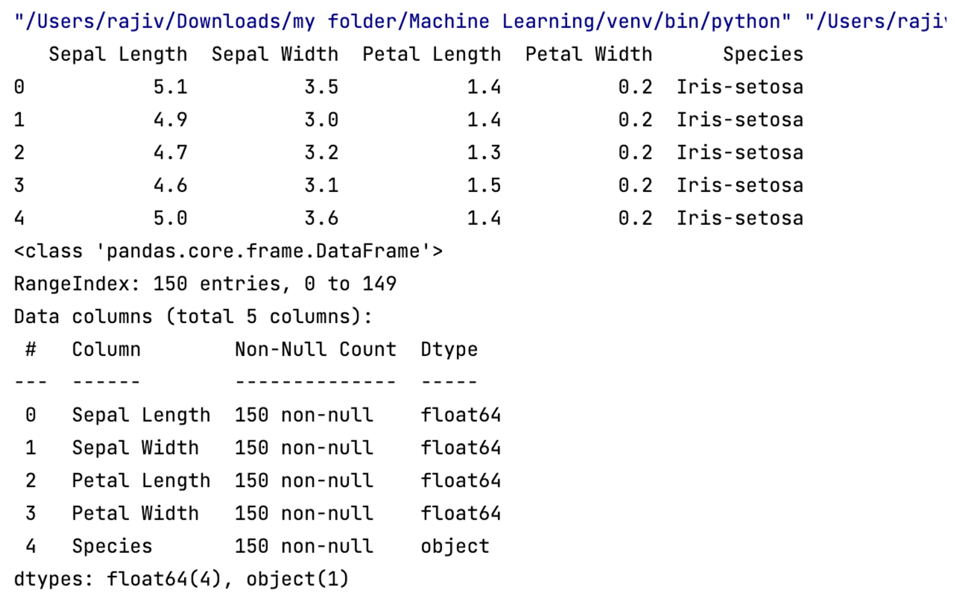
Linear Regression : Linear Regression is a machine learning algorithm (Supervised learning) where the dependency between predictor variable and dependent variable is linear with each other.
In the same case if we come across multiple variables in a dataset then we can regard it to be **multivariate regression**.
The equation for **linear regression** is given by : **y=mx+c+e ; M = slope**
C = intercept
e= error in model.

The equation for Multivariate regression is given by : **Y=BX1+BX2+BX3…+b or**
$\beta 0+ \beta 1xi1+\beta 2xi2 +...+\beta p\ xip +\varepsilon$

yi=dependent variable
X_i = explanatory variables
β0= y intercept
βp= slope coefficients for each explanatory variable

Here I initially imported all the required packages for processing the data and data visualization. Usually linear regression consists of dependent variables to be continuous hence we use multivariate linear regression with cross validation to train our model.

```
#Import Libraries for data manipulation and graph plot
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#Reading iris .csv data using pandas library
iris_dt = pd.read_csv('iris.data', sep=",", names = ["Sepal Length","Sepal Width", "Petal Length","Petal Width","Species"])

#getting head of the data to see the values and describing the dataset to understand data types
print(iris_dt.head())
print(f"\n{iris_dt.info()}")
print((f"\n{iris_dt.describe()}"))



print(f"\n{iris_dt['Species'].value_counts()}")
```



```
"/Users/rajiv/Downloads/my folder/Machine Learning/venv/bin/python" "/Users/raji
   Sepal Length  Sepal Width  Petal Length  Petal Width      Species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Sepal Length  150 non-null    float64
 1   Sepal Width   150 non-null    float64
 2   Petal Length  150 non-null    float64
 3   Petal Width   150 non-null    float64
 4   Species       150 non-null    object 
dtypes: float64(4), object(1)
```

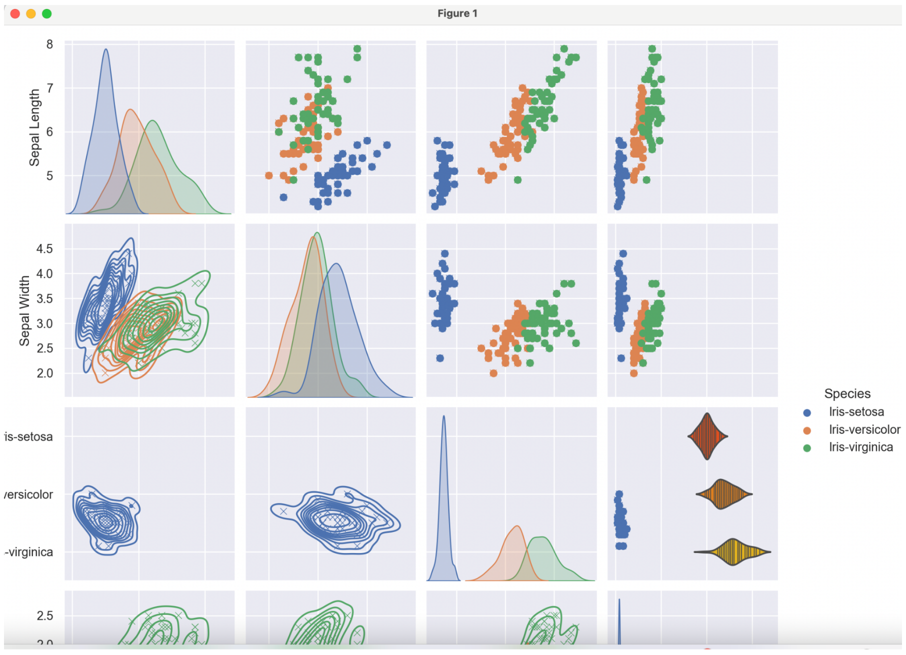
Iris dataset

Above we can clearly see that there are 6 columns , the first column is created by pandas package for easier data manipulation and the first four columns are our features i.e. sepal length, petal length, petal width, sepal width. The last column is the class label column which is basically our dependent variable. There are 3 class labels; **Iris-setosa, Iris-versicolor, and Iris-virginica.**

Data visualization:

Here I have visualized data using .map_upper and .map_lower functions from the seaborn package to analyze the data visually.

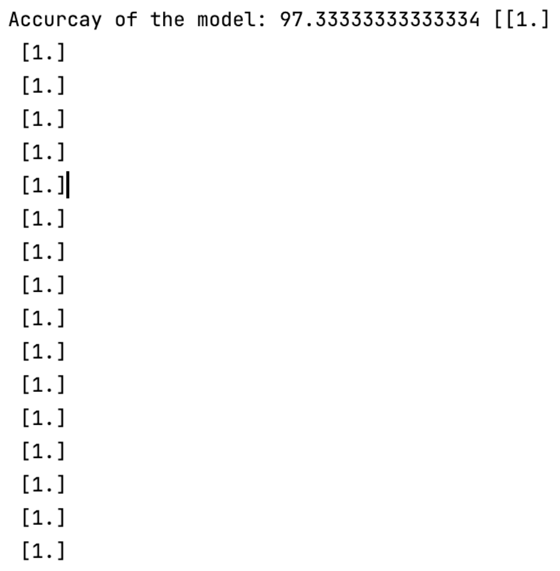
```
#datavisualisation
sns.set(rc={'figure.figsize':(8.7,6.27)})
gplot = sns.pairplot(iris_dt, hue='Species', markers='x')
gplot = gplot.map_upper(plt.scatter)
gplot = gplot.map_lower(sns.kdeplot)
sns.violinplot(x='Sepal Length', y='Species', data=iris_dt, inner='stick', palette='autumn')
plt.show()
sns.violinplot(x='Sepal Width', y='Species', data=iris_dt, inner='stick', palette='autumn')
plt.show()
sns.violinplot(x='Petal Length', y='Species', data=iris_dt, inner='stick', palette='autumn')
plt.show()
sns.violinplot(x='Petal Width', y='Species', data=iris_dt, inner='stick', palette='autumn')
plt.show()
```



```
#Model training for beta value computation
def mylinear_regression(x,y):
    np.random.seed(0)
    beta_value=np.random.randn(1,5)
    print("Beta:",beta_value)
    for i in range(itr):
        temp1=(np.dot(x,beta_value.T)-y)
        temp=np.sum(temp1 ** 2)
        arr[i]=(1/(2*x.shape[0])*temp)
        beta_value=beta_value-((a/x.shape[0])*np.dot((temp1).reshape(1,x.shape[0]),x))
    diag=plt.subplot(111)
    diag.plot(np.arange(itr),arr)
    plt.ylabel("Cost")
    plt.xlabel("Iterations")
    plt.title("RMS Error vs Alpha(0.01)")
    plt.show()
    return beta_value

#training the model and obtaining Beta Value

train_model=mylinear_regression(x,y)
predict=np.round(np.dot(x,train_model.T))
acc=(sum(predict==y)/float(len(y))*100)[0]
print("Accurcay of the model:",acc,predict) # accuracy and classification.
```

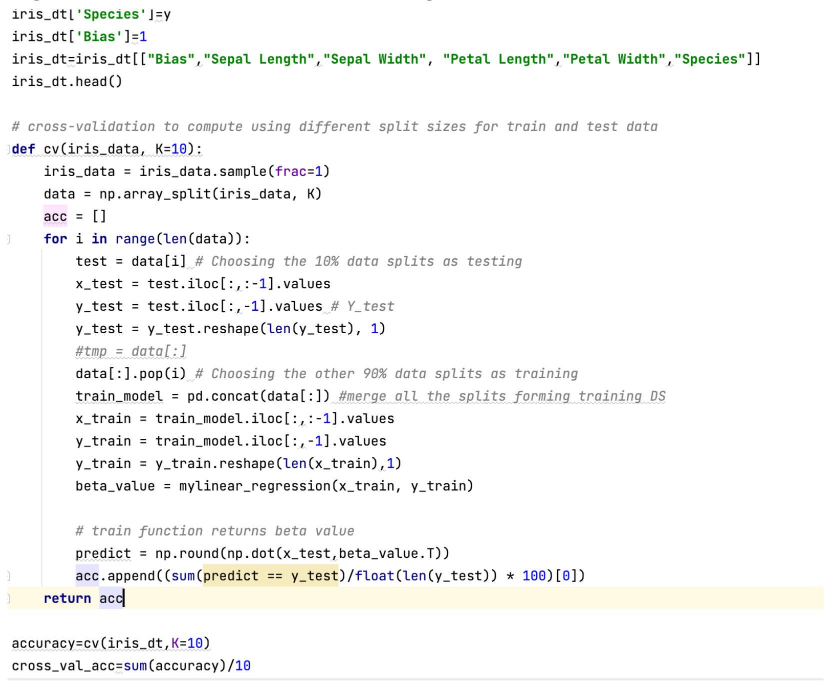
<u>**Model Training and Classification using Multivariate regression**</u>

Here we train the model by computing ß values and by applying the learning rate as 0.01, iteration= 1000 which is optimal for our solution because this allows the model to learn faster, but also that increases the cost of the sub optimal final set of weights.



```
Accurcay of the model: 97.33333333333334 [[1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
```

We see the model is predicting numerical values indicating the classification with 97.333% accuracy.

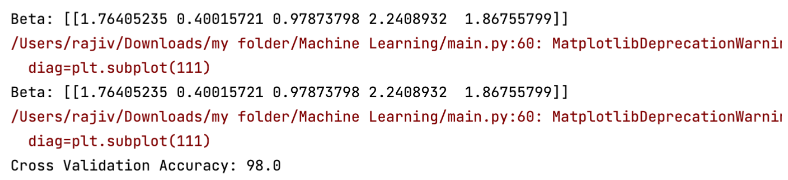


```
iris_dt['Species']=y
iris_dt['Bias']=1
iris_dt=iris_dt[["Bias","Sepal Length","Sepal Width", "Petal Length","Petal Width","Species"]]
iris_dt.head()

# cross-validation to compute using different split sizes for train and test data
def cv(iris_data, K=10):
    iris_data = iris_data.sample(frac=1)
    data = np.array_split(iris_data, K)
    acc = []
    for i in range(len(data)):
        test = data[i] # Choosing the 10% data splits as testing
        x_test = test.iloc[:,:-1].values
        y_test = test.iloc[:,-1].values # Y_test
        y_test = y_test.reshape(len(y_test), 1)
        #tmp = data[:]
        data[:].pop(i) # Choosing the other 90% data splits as training
        train_model = pd.concat(data[:]) #merge all the splits forming training DS
        x_train = train_model.iloc[:,:-1].values
        y_train = train_model.iloc[:,-1].values
        y_train = y_train.reshape(len(x_train),1)
        beta_value = mylinear_regression(x_train, y_train)

        # train function returns beta value
        predict = np.round(np.dot(x_test,beta_value.T))
        acc.append((sum(predict == y_test)/float(len(y_test)) * 100)[0])
    return acc

accuracy=cv(iris_dt,K=10)
cross_val_acc=sum(accuracy)/10
```

<u>Cross Validation</u>:

In cross validation we resample the available data; change the split ratio for test and train data, to evaluate a model on a small sample of datasets.
 I have used 10 as the value for the K- fold cross validation where we divide the data into K number of groups.

<u>Result</u>



```
Beta: [[1.76405235 0.40015721 0.97873798 2.2408932  1.86755799]]
/Users/rajiv/Downloads/my folder/Machine Learning/main.py:60: MatplotlibDeprecationWarni
  diag=plt.subplot(111)
Beta: [[1.76405235 0.40015721 0.97873798 2.2408932  1.86755799]]
/Users/rajiv/Downloads/my folder/Machine Learning/main.py:60: MatplotlibDeprecationWarni
  diag=plt.subplot(111)
Cross Validation Accuracy: 98.0
```

References

1. https://www.upgrad.com/blog/types-of-regression-models-in-machine-learning/
2. https://medium.com/analytics-vidhya/understanding-the-linear-regression-808c1f6941c0
3. https://www.kaggle.com/code/amarpandey/implementing-linear-regression-on-iris-dataset
4. https://seaborn.pydata.org/generated/seaborn.PairGrid.map_upper.html
5. https://www.kaggle.com/code/jnikhilsai/cross-validation-with-linear-regression/notebook
6. https://www.youtube.com/watch?v=4swNt7PiamQ
7. https://machinelearningmastery.com/k-fold-cross-validation/
8. https://scikit-learn.org/stable/modules/cross_validation.html