Dibyasom Puhan
SAP id: 500076104

Date: 24th August, 2020
Roll no.: R177219075

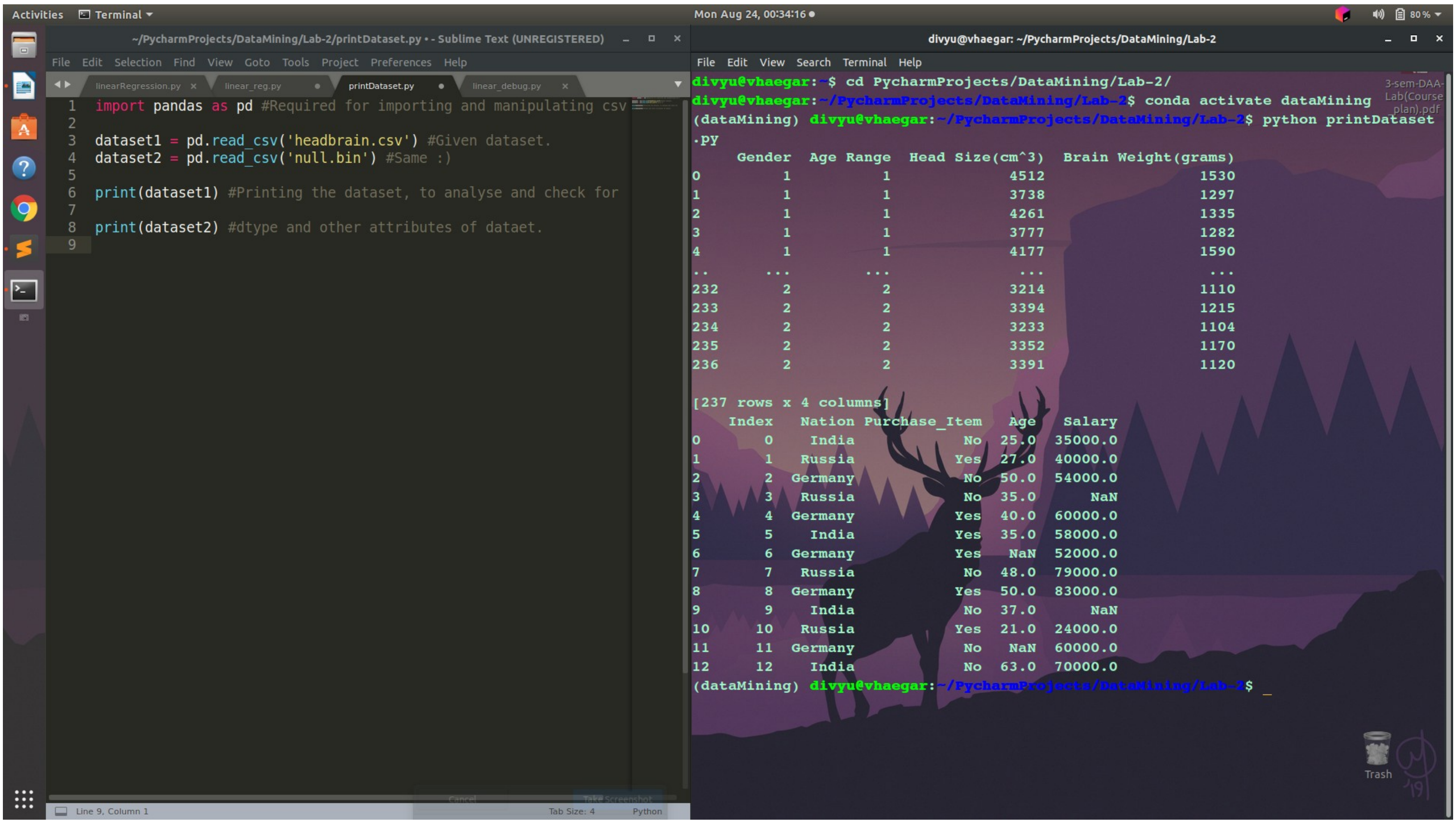
Data-Mining and predictions by machine~

~Given datasets

> headbrain -> Datasets of shape (237 x 4), the columns of significant usecase being `head size` and `brain wieght`. Provides information about brain wieght(in g) w.r.t head size(in cm^3).

> age_salary -> Dataset of size (13 X 4), the columns of significant usecase being `head size` and `brain wieght`. Provides information about salary of an individual w.r.t his/her age.

Glance of the given datasets



Dibyasom Puhan
SAP id: 500076104

Date: 24th August, 2020
Roll no.: R177219075

Which ML-model is best applicable ?

Since, data-points seem to have linear-dependance i.e can be represented as linear-algebraic combination of the known-variable, Single variable linear regression model will be best applicable.

I will be calculating Linear Regression using Least Square method, the required parameters for the same are,

> x (known variable ~ headsize / age.
> y (uknown variable ~ brain weight / salary.
> x-mean(x)
> y-mean(y)
> [x-mean(x)]^2 {Taking the square is necessary because, x-mena(x) can be negative at instances, since it's deviation from mean, sum of all data-points would be 0, hence squaring the term is necessary}
> [y-mean(y) * x-mean(x)]

The goal is to find the equation of the best-fit line that describes the relation from known to unknown variables with least error.

Coefficient (or slope of line) :

Σ{x-mean(x)]^2 / Σ{(y-mean(y) * (x-mean(x)}

Intercept (or y-intercept of line):

mean(y) – slop * mean(x)

Dibyasom Puhan **Date: 24th August, 2020**
SAP id: 500076104 **Roll no.: R177219075**

Code for the Single Variable Linear Regression model

```python
# Linear Regression Model.
# Code by,Dibyasom Puhan           24th August, 2020.
#
# Linear regression?
#~~~~~~~~~~~~~~~~~~~~~
# Linear regression is an effective ML-model to predict uknown variable when the unknown variable varies with some
# constant proportionality with the known variable(s), whether positive or negative.
#
# Using this particular piece of code,
# in main() call the class linear_reg with args <fileName>, <Known Variable>, <Unknown variable>,
# <Title for visualization>, <Estimation of missing data points, T/F ?>
# Enjoy all necessary insights about data, with beautiful visualizations.

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

class linear_reg:

    def __init__(self, fileName, Xcoloumn, Ycoloumn, title, estReq=False): #Initializes all the necessary variables to keep track of model.
        self.estimationRequired = estReq
        self.title    = title
        self.dataset  = pd.read_csv(fileName)
        self.Xcoloumn = Xcoloumn
        self.Ycoloumn = Ycoloumn
        self.yPred    = []
        self.bestFit  = {}
        self.X        = np.array([])
        self.Y        = np.array([])
        self.X_test   = np.array([])
        self.Y_test   = np.array([])
        self.yPred_all= np.array([])
        self.acc      = 0

    def preprocess(self): #Substitutes missing data in Independent column, making it ready to be used by model
        self.dataset[self.Xcoloumn].fillna(value=self.dataset[self.Xcoloumn].mean(), inplace=True)

    def split(self): #Splits the dataset for training and testing purpose
        self.X = np.array(self.dataset[self.Xcoloumn][:200])
        self.Y = np.array(self.dataset[self.Ycoloumn][:200])
        self.X_test = np.array(self.dataset[self.Xcoloumn][200:])
        self.Y_test = np.array(self.dataset[self.Ycoloumn][200:])

    def newMatrix_noNan(self): #Mimics the dataset but with no missing data.
        data = self.dataset.dropna()
        self.Y = np.array(data['Salary'])
        self.X = np.array(data['Age'])
        self.X_test = np.array([x for x in self.dataset['Age'] if x not in self.X])

    def apply_linear_regression(self): #Applies linear regression and fetches the best-fit eq parameters in dictionary.
        x_mean, y_mean = np.mean(self.X), np.mean(self.Y)
        x_x, y_y = self.X-x_mean, self.Y-y_mean
        x_ssd = np.array(np.square(x_x))
        yDiff_xDiff = np.array(np.multiply(x_x, y_y))
        coefficient = np.sum(yDiff_xDiff) / np.sum(x_ssd)
        intercept   = y_mean - coefficient*x_mean
        self.bestFit = {'coefficient': coefficient, 'intercept': intercept}

    def generate_yPred(self): #Generates predicted-Y using best-fit eq generated via linear regression.
        self.yPred = [(self.bestFit['coefficient']*x+self.bestFit['intercept']) for x in self.X_test]
        self.yPred_all = [(self.bestFit['coefficient']*x+self.bestFit['intercept']) for x in self.X]
        print(self.yPred)

    def calculate_accuracy(self): #Calculates accuracy of prediction of the model.
        y_error = [abs(y_test-y_pred)/y_test for y_test,y_pred in zip(self.Y_test, self.yPred)]
        self.acc = (100-sum(y_error)/len(y_error)*100)

    def trainAndTest(self):  #Calls all necessary functions in proper heirarchy for smooth control flow and convenience, hehe:)
        self.split()
        self.apply_linear_regression()
        self.generate_yPred()
        self.calculate_accuracy()

    def estimateMissing(self): #Calls all necessary functions in defined heirarchy to predict and fill missing values in dataset too.
        self.preprocess()
        self.newMatrix_noNan()
        self.apply_linear_regression()
        self.generate_yPred()
```

Dibyasom Puhan
SAP id: 500076104

Date: 24th August, 2020
Roll no.: R177219075

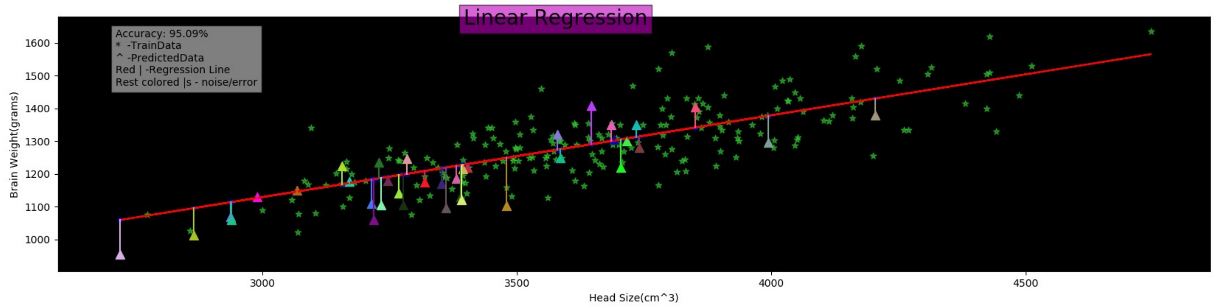
```python
def visualize(packet): #Visualizes the linear model, with regression line, predicted data-points, given data-points, and noise cloud.
    fig, axs = plt.subplots(2)

    for plot_id, self in enumerate(packet):
        keyContent = 'Accuracy: {:.2f}%'.format(self.acc)+self.title
        fig.patch.set_facecolor('xkcd:mint green')
        axs[plot_id].text(0.05, 0.95, keyContent, transform=axs[plot_id].transAxes, fontsize=10, bbox=dict(facecolor='w', alpha=0.5), verticalalignm
        axs[plot_id].set_facecolor('k')
        axs[plot_id].plot(self.X_test, self.yPred, color='red')
        axs[plot_id].plot(self.X, self.yPred_all, color ='red')
        axs[plot_id].scatter(self.X, self.Y, c='#32CD32', s=40, marker="*", alpha = '0.6')
        if self.estimationRequired:
            for x,y in zip(self.X_test, self.yPred):
                rgb = (np.random.random(), np.random.random(), np.random.random())
                axs[plot_id].scatter(x,  y, c=rgb, s=75, marker="^", alpha=1)
                axs[plot_id].text(x, y-8000, 'Estimated for missing pair:\n{:.2f} -> {:.2f}'.format(x, y), fontsize=10, bbox=dict(facecolor='w', alp
        else:
            for x,y,_y in zip(self.X_test, self.Y_test, self.yPred):
                rgb = (np.random.random(), np.random.random(), np.random.random())
                axs[plot_id].scatter(x, _y, c='b', s=20, marker="o", alpha=0.5)
                axs[plot_id].scatter(x,  y, c=rgb, s=75, marker="^", alpha=1)
                axs[plot_id].plot([x, x], [_y, y], color=rgb)

    for ax,self in zip(axs.flat, packet):
        ax.set(xlabel= self.Xcoloumn, ylabel= self.Ycoloumn)

    fig.suptitle('Linear Regression', fontsize=20, bbox=dict(facecolor='m', alpha=0.6))
    fig.tight_layout()
    plt.show()

def main(): #Main funtion, basically the linear-regression applicable dateset should be called as memeber of class linear-reg for model to initiate.
    HeadBrain  = linear_reg('headbrain.csv', 'Head Size(cm^3)', 'Brain Weight(grams)',  '\n*  -TrainData\n^ -PredictedData\nRed | -Regression Line\n
    HeadBrain.trainAndTest()
    AgeSalary  = linear_reg('null.bin', 'Age', 'Salary', 'Red | - Regression line\n* -TrainData\n^ -PredictedData', True)
    AgeSalary.estimateMissing()
    visualize([HeadBrain, AgeSalary]) #Calls visualizer to plot every insight gained from dataset.

if __name__=="__main__":
```

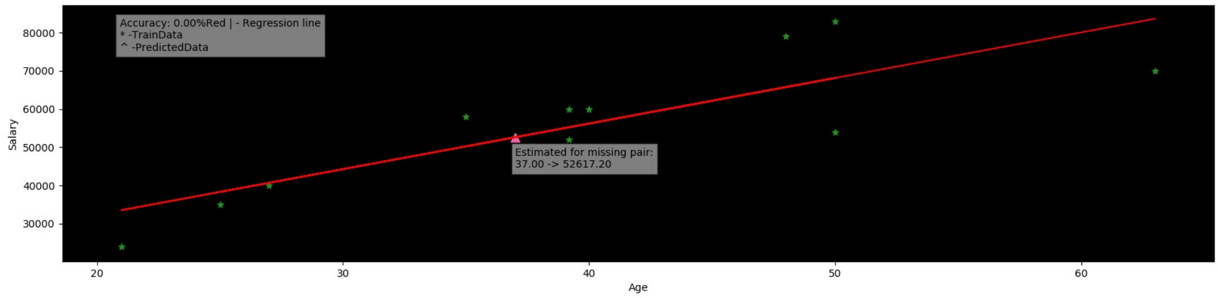
```python

if __name__=="__main__":
    main()

```

Output of the model with visualization, in next page >>

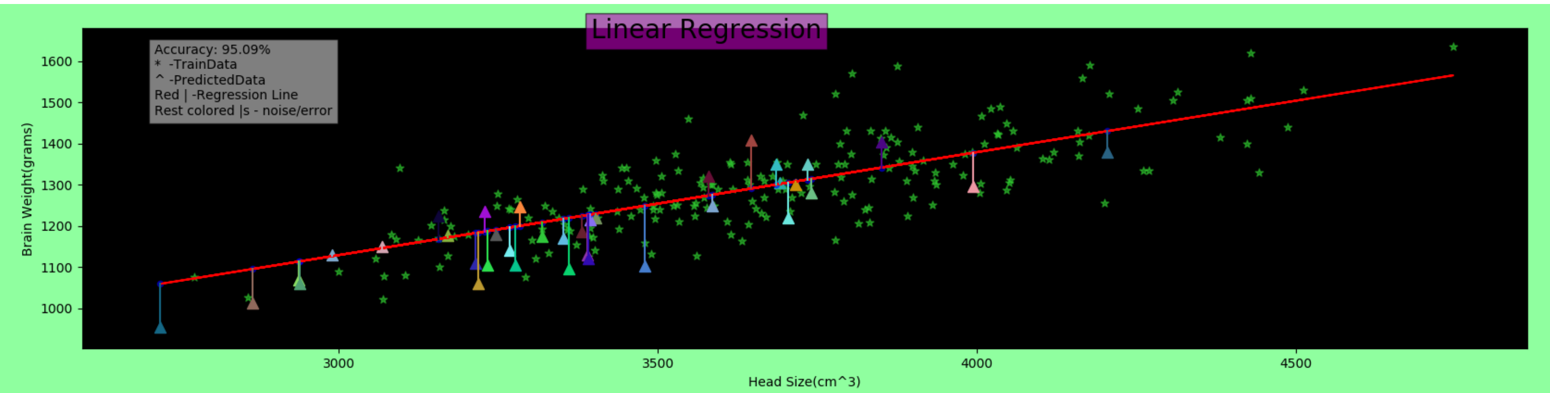
Dibyasom Puhan
SAP id: 500076104

Date: 24th August, 2020
Roll no.: R177219075





Zooming in | Predicts Brain-Weight on the basis of Head-Size.
| C O L O R F U L data-points are predicted data.
| Lines connecting them represent the error.
| Green data points are plots of data in data-set.



Zooming into age-salary