**ქვიზი 5-ის და ფინალურის ნიმუში**

**ხელით დასაწერი ფურცელზე:**

ფურცელზე დაწერილის გაფორმება: ფურცელს უნდა ეწეროს სახელი და გვარი, ასევე თარიღი, თუკი რამდენიმე ფურცელს გამოიყენებთ ფურცლები დანომროთ. საკითხები შესაბამისი ნუმერაციით გამოსახეთ. ხელნაწერი იყოს გარჩევადი ნაწერის კუთხით. ასევე მისი ფოტო ასლიც უნდა იყოს გარჩევადი. ფოტოებიც დანომრეთ.

**1. მოცემულია ვექტორები a(-1;2;0) b(3;2;1) გამოთვალეთ:**

**ა) a და b ვექტორის სიგრძე**

**ბ) b და a ვექტორების სკალარული ნამრავლი**

**გ) b და a ვექტორების ვექტორული ნამრავლი**

**ამოხსნა:**

**Text, letter

Description automatically generated**

**2. მოცემულია მატრიცა**

****

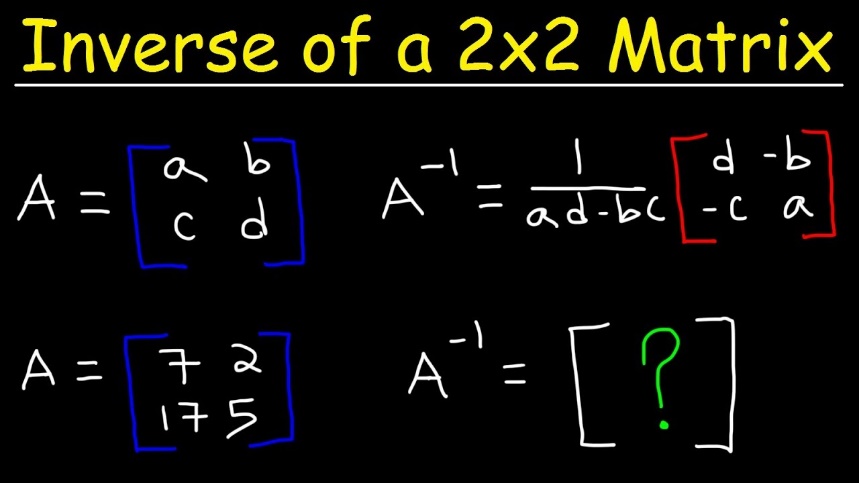
**უნდა დაითვალოთ:**

**ა) ტრანსპონირებული მატრიცა **

**ტრანსპონირებული მატრიცა მიიღება თუ არსებული მატრიცის სტრიქონებს გადავწერთ სვეტებად ანდაც მატრიცის არსებულ სვეტებს გადავწერთ სტრიქონებად.**

****

**ბ) გამოიანგარიშეთ შებრუნებული მატრიცა **



**იმისთვის რომ იპოვოთ მატრიცის შებრუნებული საჭიროა ეს მატრიცა იყოს კვადრატული და მისი დეტერმინატი იყოს ნულისაგან განსხავებული. როგორც სურათზე ხედავთ მარტივია 2x2 მატრიცის შებრუნებულის პოვნა და დაწერა. ამისათვის გვჭირდება 1 შეფარდებული დეტერმინანტთან გავამრავლოთ მიკავშირებულ მატრიცაზე. დეტერმინანტი გამოითვლება ცენტრალურ დიაგონალზე არსებულ კომპონენტთა ნამრავლს გამოკლებული ირიბ დიაგონალზე არსებულთა ნამრავლი. ხოლო მიკავშირებული მატრიცა მიიღება მოცემული მატრიცის კომპონენტებისგან, კერძოდ ცენტრალურ დიაგონალზე არსებულ ელემენტებს უბრალოდ ვუცვლით ადგილებს და ირივ დიაგონალზე არსებულ კომპონენტებს ვტოვებთ ადგილზე და ვუცვლით ნიშანს.**

****

**დ) გამოთვალეთ პირველი რიგის ნორმა**

**პირველი რიგის ნორმა რომ დავითვალოთ ამისთვის გვჭირდება მატრიცის თითოეულ სვეტში შევკრიბოთ მოდულით აღებული კომპონენტები. მიღებული ჯამები შევადაროთ ერთმანეთს და ამოვირჩიოთ მაქსიმუმი.**

**ამოხსნა:**

**A piece of paper with writing on it

Description automatically generated**

**ე) გამოთვალეთ უსუსრულო რიგის ნორმა**

**უსუსრულო ნორმა რომ დავითვალოთ ამისთვის გვჭირდება მატრიცის თითოეულ სტრიქონში შევკრიბოთ მოდულით აღებული კომპონენტები. მიღებული ჯამები შევადაროთ ერთმანეთს და ამოვირჩიოთ მაქსიმუმი.**

**ამოხსნა:**

**A piece of paper with writing on it

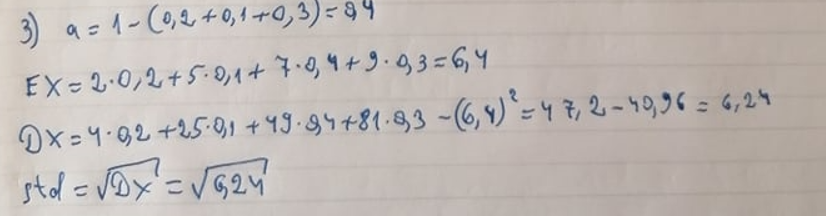
Description automatically generated**

**3. მოცემულია შემთხვევითი X დისკრეტული სიდიდის განაწილებება**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **X** | **2** | **5** | **7** | **9** |
| **Pi** | **0.2** | **0.1** | **a** | **0.3** |

**გამოთვალეთ უცნობი a ალბათობა, მათემატიკური ლოდინი, დისპერსია და სტანდარტული გადახრა.**

**ამოხსნა:**

****

**პროგრამულად შესასრულებელი:**

**შეგიძლია წარმოადგინოთ py ფაილები ან კოდები რაიმე ტექსტურ ფაილში. აქაც გამოიყენთ დავალების შესაბამისი ნუმერაცია.**

**!!!ვიზუალიზაცია მხოლოდ ქვიზზეა ფინალურში არ იქნება!!!**

**ვიზუალიზაციაა 4-9 საკითხები**

**4. მოცემულია IT საგნებზე სტუდენთა ცხრილი:**

|  |  |
| --- | --- |
| **name\_of\_class** | **students** |
| **Python** | **33** |
| **Statistics** | **27** |
| **Machine Learning** | **25** |
| **Data Science** | **39** |
| **Big Data** | **32** |

**გამოსახეთ bar დიაგრამით, დაიტანეთ y ღერზე Number of students, დიაგრამას გაუკეთეთ სათაურად Subjects enrolled by students და შენი სახელი და გვარი.**

**ამოხსნა:**

**import matplotlib.pyplot as plt**

**name\_of\_class = ['Python', 'Statistics', 'Machine Learning', 'Data Science', 'Big Data']**

**students = [33,27,25,39,32]**

**plt.bar(name\_of\_class,students)**

**plt.ylabel('Number of students')**

**plt.title('Subjects enrolled by students\nAleksandre Chakhvadze')**

**plt.show()**

**5.** **მოცემულია 2 სტუდენტის საგნების შედეგების ცხრილი:**

|  |  |  |
| --- | --- | --- |
|  | Jack | John |
| Maths | **95** | **85** |
| Statistics | **85** | **82** |
| Python | **74** | **64** |
| Data Science | **75** | **70** |
| English | **80** | **82** |

**გამოსახეთ bar დიაგრამით, დაიტანეთ y ღერზე Scores, დიაგრამას გაუკეთეთ სათაურად Scores by Students და Exam ML. Jack-ის სვეტის ფერი იყოს ლურჯი, ხოლო John-ის წითელი. სვეტის სიგანე იყოს 0.4 .**

**ამოხსნა:**

**import numpy as np**

**import matplotlib.pyplot as plt**

**scores\_Jack = ( 95 , 85 , 74 , 75 , 80 )**

**scores\_John = ( 85 , 82 , 64 , 70 , 82 )**

**fig, ax = plt.subplots ()**

**indexes = np.arange (len (scores\_Jack))**

**bar\_width = 0.4**

**data1 = plt.bar (indexes, scores\_Jack, bar\_width, color = 'b' , label = 'Jack' )**

**data2 = plt.bar (indexes + bar\_width, scores\_John, bar\_width, color = 'r' , label = 'John' )**

**plt.ylabel ( 'Scores' )**

**plt.title ( 'Scores by Students\nExam ML' )**

**plt.xticks (indexes + bar\_width / 2 , ( 'Maths' , 'Statistics' , 'Python' , 'Data Science' , 'English' ))**

**plt.legend ()**

**plt.tight\_layout ()**

**plt.show ()**

**6. მოცემულია 3 სტუდენტის საგნების შედეგების ცხრილი:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Jack | John | Lado |
| Maths | **95** | **85** | **80** |
| Statistics | **85** | **82** | **81** |
| Python | **74** | **64** | **74** |
| Data Science | **75** | **70** | **74** |
| English | **80** | **82** | **88** |

**გამოსახეთ bar დიაგრამით, დაიტანეთ y ღერზე Scores, დიაგრამას გაუკეთეთ სათაურად Scores by Students და Exam ML BTU. Jack-ის სვეტის ფერი იყოს ლურჯი, John-ის წითელი, ხოლო Lado-სი კი ყვითელი. სვეტის სიგანე იყოს 0.3 .**

**ამოხსნა:**

**import numpy as np**

**import matplotlib.pyplot as plt**

**scores\_Jack = ( 95 , 85 , 74 , 75 , 80 )**

**scores\_John = ( 85 , 82 , 64 , 70 , 82 )**

**scores\_Lado = ( 80 , 81 , 74 , 74 , 88 )**

**fig, ax = plt.subplots ()**

**indexes = np.arange (len (scores\_Jack))**

**bar\_width = 0.3**

**data1 = plt.bar (indexes, scores\_Jack, bar\_width, color = 'b' , label = 'Jack' )**

**data2 = plt.bar (indexes + bar\_width, scores\_John, bar\_width, color = 'r' , label = 'John' )**

**data3 = plt.bar (indexes + 2\*bar\_width, scores\_Lado, bar\_width, color = 'y' , label = 'Lado' )**

**plt.ylabel ( 'Scores' )**

**plt.title ( 'Scores by Students\nExam ML BTU' )**

**plt.xticks (indexes + bar\_width , ( 'Maths' , 'Statistics' , 'Python' , 'Data Science' , 'English' ))**

**plt.legend ()**

**plt.tight\_layout ()**

**plt.show ()**

**7. მოცემულია ცხრილში ამ თვეში გაყიდული ამტომანქანები:**

|  |  |
| --- | --- |
| cars | numbers\_cars |
| FORD | **13** |
| TESLA | **26** |
| JAGUAR | **39** |
| AUDI | **13** |
| BMW | **39** |
| MERCEDES | **78** |

**გამოსახეთ pie დიაგრამით პროცენტული წილების მითითებით. დიაგრამის ზომა იყოს (4;4).**

**ამოხსნა:**

**from matplotlib import pyplot as plt**

**import numpy as np**

**cars = ['FORD', 'TESLA', 'JAGUAR','AUDI', 'BMW', 'MERCEDES']**

**numbers\_cars = [13, 26, 39, 13, 39, 78]**

**fig = plt.figure(figsize =(4, 4))**

**plt.pie(numbers\_cars, labels=cars, autopct='%1.1f%%')**

**plt.title ( 'Sales\_BTU' )**

**plt.show()**

**8. მოცემულია ცხრილში ინფლაციის მაჩვენებელელი წლების განმავლობაში:**

|  |  |
| --- | --- |
| Year | inflation\_rate |
| 2000 | **2.8** |
| 2001 | **3.2** |
| 2002 | **4** |
| 2003 | **3.7** |
| 2004 | **1.2** |
| 2005 | **6.9** |
| 2006 | **7** |
| 2007 | **6.5** |
| 2008 | **6.23** |
| 2009 | **4.5** |

**გამოსახეთ plot დიაგრამით. დაიტანეთ x ღერძზე Year, დაიტანეთ y ღერძზე Inflation Rate, გაუკეთეთ სათაური Inflation Rate Vs Year. ფონტის ზომები იყოს 14. ჩანდეს უკანაფონზე ბადე. წირის ფერი იყოს წითელი.**

**ამოხსნა:**

**import matplotlib.pyplot as plt**

**Year = [2000,2001,2002,2003,2004,2005,2006,2007,2008,2009]**

**inflation\_rate = [2.8, 3.2, 4, 3.7, 1.2, 6.9, 7, 6.5, 6.23, 4.5]**

**plt.plot(Year, inflation\_rate, color='red', marker='o')**

**plt.title('Inflation Rate Vs Year', fontsize=14)**

**plt.xlabel('Year', fontsize=14)**

**plt.ylabel('Inflation Rate', fontsize=14)**

**plt.grid(True)**

**plt.show()**

**9. შემთხვევითი 1000 რიცხვის საშუალებით ააგეთ განაწილებითი ჰისტოგრამა. ფერი წითელი, ფონტის ზომა 12, დიაგრამის ზომა (8;6). დაიტანეთ x ღერძზე Value, დაიტანეთ y ღერძზე Frequency, გაუკეთეთ სათაური Normal Distribution Histogram.**

**ამოხსნა:**

**import numpy as np**

**import matplotlib.pyplot as plt**

**randomNumbers = np.random.normal(size=1000)**

**plt.figure(figsize=[8,6])**

**plt.hist(randomNumbers, width = 0.5, color='r',alpha=1)**

**plt.grid(axis='y', alpha=0.5)**

**plt.xlabel('Value',fontsize=12)**

**plt.ylabel('Frequency',fontsize=12)**

**plt.title('Normal Distribution Histogram',fontsize=12)**

**plt.show()**

**!!! მოდელის აგების დატრენინგების და შეფასების ნაწილში აქ ქვიზზე ყველა განხილული ვარიანტია მოცემული ერთად**

**თუმცა გამოცდაზე ბილეთში მხოლოდ 2 მოდელი შეგხვდებათ 1 აუცილებლად იქნება წრფივი და იქნება რაღაც მეორე სხვა დანარჩენი ალტერნატივიდან**

**ყველაფრის იმპლემენტაცია სრულადაა წარმოდგენილი მე10 საკითხში.**

**ზოგადი ანუ კოდის ზედა ნაწილი იქნება ყველა მოდელისთვის საერთო. კონკრეტული მოდელი და მისი შეფასება მიკომენტარებითაა გამოყოფილი ადვილადაც შეამჩნევთ.**

**აქ არ არის წარმოდგენილი თუ კონკრეტულ დატასეტს რა სვეტები აქვს, გასაიოლებლად გამოცდაზე ეგეც იქნება მითიტებული ასევე, თუ რომელი სვეტია დამოკიდებული (ასახსნელი ანუ ლებელი) და რომელი სვტებია დამოუკიდბელები (ამხსნელები, ფიუჩერები, გახასიათებლები).**

**ასევე გამოცდაზე შეგხვდებათ მოთხოვნა გამოტანა უნდა დაიწყოს სტანდარტული გაფორმებით, როგორიცაა მაგალითად:**

**„Modelis gatesva**

**tariRi: 31.12.2022**

**student: Aleksandre Chakhvadze**

**jgufi: G1**

**varianti: 1“**

**სადაც ჩანს შესრულების თარიღი, შენი სახელი და გვარი, ჯგუფის ნომერი და ვარფიანტი.**

**სტანდარტული გაფორმების არ ქონა საკმაოდ დაგაკლებს ქულას ამ საკითხში.**

**მე11 საკითხში უბრალოდ წრფივი მოდელის კროს ვალიდაცია უნდა მოახდინოთ 5 ფოლდისთვის. წრფივი იმიტო ზოგ ვარიანტში კდელს სატრენინგოდ დიდი დრო უნდა ამ წრფივს კი მიზერული დრო უნდა.**

**10. დაწერეთ კოდი რომელიც შექმნის seaborn-ის dataset diamonds-ზე დაყრდნობით შემდეგ მოდელებს: წრფივი რეგრესია, k უახლოესი მეზობლის რეგრესია, random forest და svm.**

**დამოკიდებულ ცვლადად აიღეთ ფასი, ხოლო სხვა დანარჩენი კი დამოუკიდებელ ფაქტორებად. მოახდინეთ კატეგორიული სვეტების გადაყვანა რიცხვითში. დაყავით მოდელი 80% სატრენინგოდ დანარჩენი 20% კი სატესტოდ. მოახდინეთ მონაცემების მაშტაბირება და დაასტანდარტულეთ.**

**მიღებული მოდელები შეაფასეთ შემდეგი მეტრიკების მეშვეობით R2, საშუალო აბსოლუტური გადახრა, საშუალო კვადრატული გადახრა და ფესვი საშუალო კვადრატული გადახრიდან.**

**ამოხსნა:**

**#general for all models**

**import pandas as pd**

**import numpy as np**

**import seaborn as sns**

**diamonds\_df = sns.load\_dataset("diamonds")**

**X = diamonds\_df.drop(['price'], axis=1)**

**y = diamonds\_df["price"]**

**numerical = X.drop(['cut', 'color', 'clarity'], axis = 1)**

**categorical = X.filter(['cut', 'color', 'clarity'])**

**cat\_numerical = pd.get\_dummies(categorical,drop\_first=True)**

**X = pd.concat([numerical, cat\_numerical], axis = 1)**

**from sklearn.model\_selection import train\_test\_split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=0)**

**from sklearn.preprocessing import StandardScaler**

**sc = StandardScaler()**

**X\_train = sc.fit\_transform(X\_train)**

**X\_test = sc.transform (X\_test)**

**# LinearRegression Model**

**from sklearn.linear\_model import LinearRegression**

**lin\_reg = LinearRegression()**

**regressor = lin\_reg.fit(X\_train, y\_train)**

**y\_pred = regressor.predict(X\_test)**

**from sklearn import metrics**

**print('LinearRegression:')**

**print('R^2:', metrics.r2\_score(y\_test, y\_pred))**

**print('Mean Absolute Error:', metrics.mean\_absolute\_error(y\_test, y\_pred))**

**print('Mean Squared Error:', metrics.mean\_squared\_error(y\_test, y\_pred))**

**print('Root Mean Squared Error:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))**

**# KNeighborsRegressor Model**

**from sklearn.neighbors import KNeighborsRegressor**

**knn\_reg = KNeighborsRegressor(n\_neighbors=5)**

**regressor = knn\_reg.fit(X\_train, y\_train)**

**y\_pred = regressor.predict(X\_test)**

**from sklearn import metrics**

**print('KNeighborsRegressor:')**

**print('R^2:', metrics.r2\_score(y\_test, y\_pred))**

**print('Mean Absolute Error:', metrics.mean\_absolute\_error(y\_test, y\_pred))**

**print('Mean Squared Error:', metrics.mean\_squared\_error(y\_test, y\_pred))**

**print('Root Mean Squared Error:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))**

**# RandomForestRegressor Model**

**from sklearn.ensemble import RandomForestRegressor**

**rf\_reg = RandomForestRegressor(random\_state=42, n\_estimators=500)**

**regressor = rf\_reg.fit(X\_train, y\_train)**

**y\_pred = regressor.predict(X\_test)**

**from sklearn import metrics**

**print('RandomForestRegressor:')**

**print('R^2:', metrics.r2\_score(y\_test, y\_pred))**

**print('Mean Absolute Error:', metrics.mean\_absolute\_error(y\_test, y\_pred))**

**print('Mean Squared Error:', metrics.mean\_squared\_error(y\_test, y\_pred))**

**print('Root Mean Squared Error:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))**

**# SVR Model**

**from sklearn import svm**

**svm\_reg = svm.SVR()**

**regressor = svm\_reg.fit(X\_train, y\_train)**

**y\_pred = regressor.predict(X\_test)**

**from sklearn import metrics**

**print('SVM Regressor:')**

**print('R^2:', metrics.r2\_score(y\_test, y\_pred))**

**print('Mean Absolute Error:', metrics.mean\_absolute\_error(y\_test, y\_pred))**

**print('Mean Squared Error:', metrics.mean\_squared\_error(y\_test, y\_pred))**

**print('Root Mean Squared Error:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))**

**11. დაწერეთ კოდი რომელიც შექმნის seaborn-ის dataset diamonds-ზე დაყრდნობით შემდეგ წრფივი რეგრესიის მოდელი.**

**დამოკიდებულ ცვლადად აიღეთ ფასი, ხოლო სხვა დანარჩენი კი დამოუკიდებელ ფაქტორებად. მოახდინეთ კატეგორიული სვეტების გადაყვანა რიცხვითში. დაყავით მოდელი 80% სატრენინგოდ დანარჩენი 20% კი სატესტოდ. მოახდინეთ მონაცემების მასშტაბირება და დაასტანდარტულეთ. მოახდინეთ K fold-ირება.**

**ამოხსნა:**

**import pandas as pd**

**import numpy as np**

**import seaborn as sns**

**diamonds\_df = sns.load\_dataset("diamonds")**

**X = diamonds\_df.drop(['price'], axis=1)**

**y = diamonds\_df["price"]**

**numerical = X.drop(['cut', 'color', 'clarity'], axis = 1)**

**categorical = X.filter(['cut', 'color', 'clarity'])**

**cat\_numerical = pd.get\_dummies(categorical,drop\_first=True)**

**X = pd.concat([numerical, cat\_numerical], axis = 1)**

**from sklearn.model\_selection import train\_test\_split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=0)**

**from sklearn.preprocessing import StandardScaler**

**sc = StandardScaler()**

**X\_train = sc.fit\_transform(X\_train)**

**X\_test = sc.transform (X\_test)**

**from sklearn.linear\_model import LinearRegression**

**lin\_reg = LinearRegression()**

**regressor = lin\_reg.fit(X\_train, y\_train)**

**y\_pred = regressor.predict(X\_test)**

**from sklearn import metrics**

**print('LinearRegression:')**

**print('R^2:', metrics.r2\_score(y\_test, y\_pred))**

**print('Mean Absolute Error:', metrics.mean\_absolute\_error(y\_test, y\_pred))**

**print('Mean Squared Error:', metrics.mean\_squared\_error(y\_test, y\_pred))**

**print('Root Mean Squared Error:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))**

**from sklearn.model\_selection import cross\_val\_score**

**print(cross\_val\_score(regressor, X, y, cv=5, scoring ="neg\_mean\_absolute\_error"))**

**12. ამ საკითხში იქნება ასაგები ორფაქტორიანი მოდელი შესწავლილი რომელიმე linear ალგორითმის მიხედვით. ყურადღებით დააკვირდით რომელი ალგორითმით გთხოვენ გამოანგარიშებას ფინალურის ბილეთის ვარიანტში. აქ ყველაა წარმოდგენილი!!!!**

**მოცემული ცხრილის მიხედვით ააგეთ ორ ფაქტორიანი წრფივი მოდელი**

|  |  |  |
| --- | --- | --- |
| **X1** | **X2** | **Y** |
| **0.1** | **0.5** | **3.7** |
| **0.5** | **0.2** | **4.2** |
| **1.2** | **2.4** | **7.9** |
| **3.1** | **4.2** | **12.9** |

**გამოიანგარიშეთ დეტერმინაციის კოეფიციენტი და ასევე მოდელის ყველა კოეფიციენტი.**

**ა) მოცემული ცხრილის მიხედვით ააგეთ ორფაქტორიანი მოდელი სტანდარტული წრფივი ალგორითმით.**

**ამოხსნა:**

**#sadziebelia aseti funqcia Y=k0+k1x1+k2x2**

**import numpy as np**

**from sklearn.linear\_model import LinearRegression**

**x =np.array([**

**[0.1,0.5],[0.5,0.2],[1.2,2.4],[3.1,4.2]**

**])**

**y =np.array([3.7,4.2,7.9,12.9])**

**model = LinearRegression()**

**model.fit(x,y)**

**r\_sq =model.score(x,y)**

**print('coefficient of determination:', r\_sq)**

**print('intercept:', model.intercept\_)**

**print('slope:', model.coef\_)**

**y\_pred =model.predict(x)**

**print('predicted y is :' , y\_pred)**

**შედეგი:**

**coefficient of determination: 0.9993731429272054**

**intercept: 3.0889785870324564**

**slope: [1.6198727 1.15160403]**

**predicted y is : [ 3.82676787 4.12923574 7.7966755 12.94732088]**

**ბ) მოცემული ცხრილის მიხედვით ააგეთ ორფაქტორიანი მოდელი Ridge ალგორითმით თუ ალფა იქნება 1.4**

ამოხსნა:

**import numpy as np**

**from sklearn.linear\_model import Ridge**

**x =np.array([**

**[0.1,0.5],[0.5,0.2],[1.2,2.4],[3.1,4.2]**

**])**

**y =np.array([3.7,4.2,7.9,12.9])**

**model = Ridge(alpha=1.4)**

**model.fit(x,y)**

**r\_sq =model.score(x,y)**

**print('coefficient of determination:', r\_sq)**

**print('intercept:', model.intercept\_)**

**print('slope:', model.coef\_)**

**y\_pred =model.predict(x)**

**print('predicted y is :' , y\_pred)**

**გ) მოცემული ცხრილის მიხედვით ააგეთ ორფაქტორიანი მოდელი Lasso ალგორითმით თუ ალფა იქნება 0.5**

**ამოხსნა:**

**import numpy as np**

**from sklearn.linear\_model import Lasso**

**x =np.array([**

**[0.1,0.5],[0.5,0.2],[1.2,2.4],[3.1,4.2]**

**])**

**y =np.array([3.7,4.2,7.9,12.9])**

**model =Lasso(alpha=0.5)**

**model.fit(x,y)**

**r\_sq =model.score(x,y)**

**print('coefficient of determination:', r\_sq)**

**print('intercept:', model.intercept\_)**

**print('slope:', model.coef\_)**

**y\_pred =model.predict(x)**

**print('predicted y is :' , y\_pred)**

**დ) მოცემული ცხრილის მიხედვით ააგეთ ორფაქტორიანი მოდელი ElasticNet ალგორითმით თუ ალფა იქნება 1.0, ხოლო l1\_ratio იქნება 0.5**

**ამოხსნა:**

**import numpy as np**

**from sklearn.linear\_model import ElasticNet**

**x =np.array([**

**[0.1,0.5],[0.5,0.2],[1.2,2.4],[3.1,4.2]**

**])**

**y =np.array([3.7,4.2,7.9,12.9])**

**model =ElasticNet(alpha=1.0,l1\_ratio=0.5)**

**model.fit(x,y)**

**r\_sq =model.score(x,y)**

**print('coefficient of determination:', r\_sq)**

**print('intercept:', model.intercept\_)**

**print('slope:', model.coef\_)**

**y\_pred =model.predict(x)**

**print('predicted y is :' , y\_pred)**

**მე13 საკითხში აქ statmodels ბიბლიოთეკის გამოყენებით მოდელის აგება და შეფასებაა. გამოცდაზე შემფასებელი პარამეტრები და რეპორტი ეს სიტყვები მიგანიშნებს რო ეს გჭირდება.**

**13. ვთქვათ მოცემული გაქვსთ შემდეგი მასივები ააგეთ უმცირეს კვადრატთა მეთოდით წრფივი რეგრესია და გამოიტანეთ მოდელის შემფასებელი პარამეტრები და რეპორტი.**

**x =np.array([**

**[0.1,0.5],[0.5,0.2],[1.2,2.4],[3.1,4.2],[4.4,5.3],[1.3,4.5],[2.3,5.6],**

**[7.7,7.9]**

**])**

**y =np.array([3.7,4.2,7.9,12.9,14.8,19.0,21.6,34.5])**

**ამოხსნა:**

**import numpy as np**

**import statsmodels.api as sm**

**x =np.array([**

**[0.1,0.5],[0.5,0.2],[1.2,2.4],[3.1,4.2],[4.4,5.3],[1.3,4.5],[2.3,5.6],[7.7,7.9]**

**])**

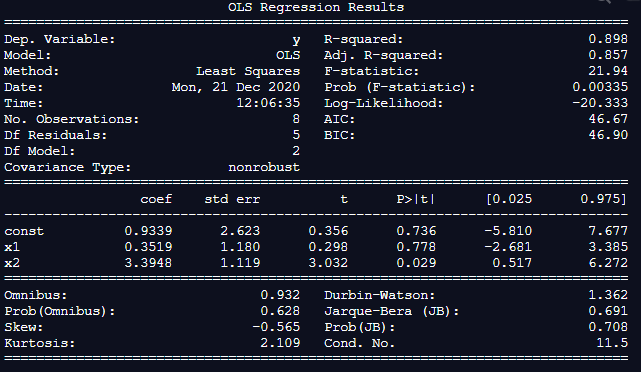
**y =np.array([3.7,4.2,7.9,12.9,14.8,19.0,21.6,34.5])**

**x =sm.add\_constant(x)**

**model = sm.OLS(y, x)**

**results = model.fit()**

**print(results.summary())**

შედეგი: