Week 1 assignment  
#advanced python

#program to triple each number of the list using map

List = {1,2,3,4,10,123,22}

new\_list = list(map(lambda x:x\*3,List))

print(new\_list)

#output [3,6,9,12,30,66,369]

x=lambda a,b:a+b

print("sum = ",x(20,10))

#output: sum = 30  
# def upperc(function):

# def wrapper():

# f=function()

# mu=f.upper()

# return mu

# return wrapper

# d@upperc

# def sayhi():

# return "hello there"

# sayhi()

# dec=upperc(sayhi)

# dec()

# l=[1,2,3,4]

# i=iter(l)

# for k in range(len(l)):

# print(next(i))

# def f():

# n=1

# print("first")

# yield n

# n=2

# print("sc")

# yield n

# def k(max=0):

# n=0

# while n<max:

# yield n\*\*2

# n+=1

# for i in k(5):

# print(i)

#tensorflow and tensorboard and plots in notbook

{"cells":[{"metadata":{"\_uuid":"8f2839f25d086af736a60e9eeb907d3b93b6e0e5","\_cell\_guid":"b1076dfc-b9ad-4769-8c92-a6c4dae69d19","trusted":true},"cell\_type":"code","source":"# This Python 3 environment comes with many helpful analytics libraries installed\n# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python\n# For example, here's several helpful packages to load in \n\nimport numpy as np # linear algebra\nimport pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)\n\n# Input data files are available in the \"../input/\" directory.\n# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory\n\nimport os\nfor dirname, \_, filenames in os.walk('/kaggle/input'):\n for filename in filenames:\n print(os.path.join(dirname, filename))\n\n# Any results you write to the current directory are saved as output.","execution\_count":114,"outputs":[]},{"metadata":{"\_uuid":"d629ff2d2480ee46fbb7e2d37f6b5fab8052498a","\_cell\_guid":"79c7e3d0-c299-4dcb-8224-4455121ee9b0","trusted":true},"cell\_type":"code","source":"import tensorflow as tf","execution\_count":115,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"ms=tf.constant([[1,2,3],[1,2,3]])","execution\_count":116,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"print(ms.shape)","execution\_count":117,"outputs":[{"output\_type":"stream","text":"(2, 3)\n","name":"stdout"}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"mk=tf.constant([[1,2],[3,4]],dtype=tf.int32)","execution\_count":118,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"mk1=tf.constant([[3,4],[5,6]],dtype=tf.int32)","execution\_count":119,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"ta=tf.add(mk,mk1)","execution\_count":120,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"print(ta)","execution\_count":121,"outputs":[{"output\_type":"stream","text":"tf.Tensor(\n[[ 4 6]\n [ 8 10]], shape=(2, 2), dtype=int32)\n","name":"stdout"}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"tc=tf.constant([[1,2],[2,3]])","execution\_count":122,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"tv=tf.get\_variable(\"tv\",dtype=tf.int32)","execution\_count":123,"outputs":[{"output\_type":"error","ename":"AttributeError","evalue":"module 'tensorflow' has no attribute 'get\_variable'","traceback":["\u001b[0;31m---------------------------------------------------------------------------\u001b[0m","\u001b[0;31mAttributeError\u001b[0m Traceback (most recent call last)","\u001b[0;32m<ipython-input-123-417cee7e5c78>\u001b[0m in \u001b[0;36m<module>\u001b[0;34m\u001b[0m\n\u001b[0;32m----> 1\u001b[0;31m \u001b[0mtv\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0mtf\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mget\_variable\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0;34m\"tv\"\u001b[0m\u001b[0;34m,\u001b[0m\u001b[0mdtype\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0mtf\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mint32\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0m","\u001b[0;31mAttributeError\u001b[0m: module 'tensorflow' has no attribute 'get\_variable'"]}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"tc1=tf.constant([[1,2],[2,3]])","execution\_count":null,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"tc2=tf.constant([[1,2],[2,3]])","execution\_count":null,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"m=tf.multiply(tc1,tc2)","execution\_count":null,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"s=tf.Session()","execution\_count":null,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"pip install tensorflow","execution\_count":null,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"pip install tensorflow version== 1.14","execution\_count":null,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"raw","source":"import tensorflow as tf\ntv=tf.get\_variable(\"tv\",dtype=tf.int32,initializer=tc1)\nprint(tv)"},{"metadata":{"trusted":true},"cell\_type":"code","source":"tf.\_\_version\_\_","execution\_count":124,"outputs":[{"output\_type":"execute\_result","execution\_count":124,"data":{"text/plain":"'2.1.0'"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"import numpy as np","execution\_count":125,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"x=np.random.sample((100,2))","execution\_count":126,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"print(x)","execution\_count":127,"outputs":[{"output\_type":"stream","text":"[[0.32105101 0.28511851]\n [0.63886036 0.16815084]\n [0.36522177 0.75859433]\n [0.08683655 0.72447578]\n [0.9348041 0.4314334 ]\n [0.09375693 0.76646444]\n [0.36945124 0.36038755]\n [0.64907773 0.11291121]\n [0.22237389 0.50149169]\n [0.32227702 0.63065982]\n [0.43132338 0.2556314 ]\n [0.56944909 0.03101038]\n [0.66825012 0.96209779]\n [0.06616398 0.00996935]\n [0.77556484 0.6031201 ]\n [0.81240333 0.17368997]\n [0.37358154 0.01926862]\n [0.91391884 0.6768733 ]\n [0.16736936 0.61598411]\n [0.5103634 0.09799164]\n [0.85205945 0.40158486]\n [0.53978433 0.25289111]\n [0.70489144 0.61660719]\n [0.19855799 0.51693413]\n [0.52791971 0.76003633]\n [0.81504779 0.51999077]\n [0.11767045 0.80070006]\n [0.79772894 0.45862959]\n [0.61104696 0.7646428 ]\n [0.46325794 0.55326212]\n [0.60385438 0.20351354]\n [0.35154823 0.62996188]\n [0.98250424 0.51562871]\n [0.43419152 0.64326634]\n [0.50017256 0.91645122]\n [0.12603961 0.85648598]\n [0.25956826 0.00827751]\n [0.33942377 0.07774153]\n [0.32473456 0.64315251]\n [0.41999035 0.21394021]\n [0.87033153 0.36257827]\n [0.44900281 0.9985632 ]\n [0.78018127 0.22740979]\n [0.66528194 0.32252285]\n [0.83776398 0.06007636]\n [0.51516992 0.98382661]\n [0.40862182 0.29625001]\n [0.88926724 0.76835046]\n [0.9001972 0.68267355]\n [0.20818561 0.46246411]\n [0.76821996 0.89968907]\n [0.09657704 0.67709955]\n [0.64119381 0.68173453]\n [0.22146987 0.950886 ]\n [0.77586129 0.92735428]\n [0.65217852 0.21553639]\n [0.79379782 0.91520786]\n [0.55504631 0.96412947]\n [0.90606944 0.52441231]\n [0.50139456 0.79200402]\n [0.31267307 0.76211899]\n [0.50624926 0.1798437 ]\n [0.54240858 0.12594588]\n [0.93098341 0.3162046 ]\n [0.40119092 0.422876 ]\n [0.04302396 0.19199781]\n [0.06635237 0.86006926]\n [0.40157279 0.64440617]\n [0.12920546 0.57852759]\n [0.5273469 0.93219647]\n [0.49128962 0.43712548]\n [0.24250476 0.07848423]\n [0.80059743 0.73019993]\n [0.47658108 0.21730092]\n [0.84336751 0.65317537]\n [0.29107659 0.36914201]\n [0.36651005 0.5673241 ]\n [0.71596018 0.47281824]\n [0.11482098 0.20542377]\n [0.85843438 0.88981561]\n [0.69140474 0.63661782]\n [0.87302285 0.38513176]\n [0.63070963 0.33477887]\n [0.50396466 0.12742938]\n [0.96147736 0.72359938]\n [0.89429144 0.40951144]\n [0.00198676 0.6116162 ]\n [0.03562613 0.42539294]\n [0.43593263 0.1666723 ]\n [0.21832858 0.40314213]\n [0.38603499 0.58434848]\n [0.86736001 0.45180173]\n [0.56223378 0.96397923]\n [0.50465107 0.07422716]\n [0.87717869 0.27320068]\n [0.04799041 0.54656907]\n [0.26018265 0.93053354]\n [0.79382547 0.04797211]\n [0.76251116 0.64575818]\n [0.65449156 0.89630063]]\n","name":"stdout"}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"dataset = tf.data.Dataset.from\_tensor\_slices(x)\n# create the iterator\niter = dataset.make\_one\_shot\_iterator()\nel = iter.get\_next()\nwith tf.Session() as sess:\n print(sess.run(el)) \n# output: [ 0.42116176 0.40666069]","execution\_count":128,"outputs":[{"output\_type":"error","ename":"AttributeError","evalue":"'TensorSliceDataset' object has no attribute 'make\_one\_shot\_iterator'","traceback":["\u001b[0;31m---------------------------------------------------------------------------\u001b[0m","\u001b[0;31mAttributeError\u001b[0m Traceback (most recent call last)","\u001b[0;32m<ipython-input-128-37a78740c9c5>\u001b[0m in \u001b[0;36m<module>\u001b[0;34m\u001b[0m\n\u001b[1;32m 1\u001b[0m \u001b[0mdataset\u001b[0m \u001b[0;34m=\u001b[0m \u001b[0mtf\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mdata\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mDataset\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mfrom\_tensor\_slices\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mx\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m 2\u001b[0m \u001b[0;31m# create the iterator\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0;32m----> 3\u001b[0;31m \u001b[0miter\u001b[0m \u001b[0;34m=\u001b[0m \u001b[0mdataset\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mmake\_one\_shot\_iterator\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0m\u001b[1;32m 4\u001b[0m \u001b[0mel\u001b[0m \u001b[0;34m=\u001b[0m \u001b[0miter\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mget\_next\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m 5\u001b[0m \u001b[0;32mwith\u001b[0m \u001b[0mtf\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mSession\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0;34m)\u001b[0m \u001b[0;32mas\u001b[0m \u001b[0msess\u001b[0m\u001b[0;34m:\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n","\u001b[0;31mAttributeError\u001b[0m: 'TensorSliceDataset' object has no attribute 'make\_one\_shot\_iterator'"]}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"import tensorflow as tf\nimport numpy as np\nEPOCHS = 10\nBATCH\_SIZE = 15","execution\_count":null,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"features, labels = (np.array([np.random.sample((100,2))]), \nnp.array([np.random.sample((100,1))]))\ndataset = tf.data.Dataset.from\_tensor\_slices((features,labels)).repeat().batch(BATCH\_SIZE)","execution\_count":null,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"iter = dataset.make\_one\_shot\_iterator()\nx, y = iter.get\_next()","execution\_count":null,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"tf.compat.v1.data.Iterator(dataset)","execution\_count":null,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"import numpy as np\nimport pandas as pd\n \nimport tensorflow as tf\nfrom tensorflow import keras\nfrom tensorflow.keras import layers","execution\_count":null,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"data={'Age':[20,24,38,39,24,36,28,29,30,20,24,40,30,26,30,23,29,30,19,30],'Profession':[1,3,7,4,5,2,2,1,4,5,7,3,5,6,6,1,2,4,3,5],\n 'Day':[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],\n'number\_of\_husbands':[1,3,2,5,10,8,13,15,20,26,24,20,21,19,28,18,20,28,29,30]}\n \ndataset=pd.DataFrame(data,columns=['Age','Profession','day',\n'number\_of\_husbands'])","execution\_count":129,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"print(dataset)","execution\_count":130,"outputs":[{"output\_type":"stream","text":" Age Profession day number\_of\_husbands\n0 20 1 NaN 1\n1 24 3 NaN 3\n2 38 7 NaN 2\n3 39 4 NaN 5\n4 24 5 NaN 10\n5 36 2 NaN 8\n6 28 2 NaN 13\n7 29 1 NaN 15\n8 30 4 NaN 20\n9 20 5 NaN 26\n10 24 7 NaN 24\n11 40 3 NaN 20\n12 30 5 NaN 21\n13 26 6 NaN 19\n14 30 6 NaN 28\n15 23 1 NaN 18\n16 29 2 NaN 20\n17 30 4 NaN 28\n18 19 3 NaN 29\n19 30 5 NaN 30\n","name":"stdout"}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"dataset['Profession'] = dataset['Profession'].map(lambda x: {1: 'doctor', 2:'engineer',3:'dancer',4:'painter',5:'singer',6:'lawyer',7:'artist'}.get(x))\ndataset = pd.get\_dummies(dataset, prefix='', prefix\_sep='')\ndataset.head()","execution\_count":131,"outputs":[{"output\_type":"execute\_result","execution\_count":131,"data":{"text/plain":" Age number\_of\_husbands artist dancer doctor engineer lawyer painter \\\n0 20 1 0 0 1 0 0 0 \n1 24 3 0 1 0 0 0 0 \n2 38 2 1 0 0 0 0 0 \n3 39 5 0 0 0 0 0 1 \n4 24 10 0 0 0 0 0 0 \n\n singer \n0 0 \n1 0 \n2 0 \n3 0 \n4 1 ","text/html":"<div>\n<style scoped>\n .dataframe tbody tr th:only-of-type {\n vertical-align: middle;\n }\n\n .dataframe tbody tr th {\n vertical-align: top;\n }\n\n .dataframe thead th {\n text-align: right;\n }\n</style>\n<table border=\"1\" class=\"dataframe\">\n <thead>\n <tr style=\"text-align: right;\">\n <th></th>\n <th>Age</th>\n <th>number\_of\_husbands</th>\n <th>artist</th>\n <th>dancer</th>\n <th>doctor</th>\n <th>engineer</th>\n <th>lawyer</th>\n <th>painter</th>\n <th>singer</th>\n </tr>\n </thead>\n <tbody>\n <tr>\n <th>0</th>\n <td>20</td>\n <td>1</td>\n <td>0</td>\n <td>0</td>\n <td>1</td>\n <td>0</td>\n <td>0</td>\n <td>0</td>\n <td>0</td>\n </tr>\n <tr>\n <th>1</th>\n <td>24</td>\n <td>3</td>\n <td>0</td>\n <td>1</td>\n <td>0</td>\n <td>0</td>\n <td>0</td>\n <td>0</td>\n <td>0</td>\n </tr>\n <tr>\n <th>2</th>\n <td>38</td>\n <td>2</td>\n <td>1</td>\n <td>0</td>\n <td>0</td>\n <td>0</td>\n <td>0</td>\n <td>0</td>\n <td>0</td>\n </tr>\n <tr>\n <th>3</th>\n <td>39</td>\n <td>5</td>\n <td>0</td>\n <td>0</td>\n <td>0</td>\n <td>0</td>\n <td>0</td>\n <td>1</td>\n <td>0</td>\n </tr>\n <tr>\n <th>4</th>\n <td>24</td>\n <td>10</td>\n <td>0</td>\n <td>0</td>\n <td>0</td>\n <td>0</td>\n <td>0</td>\n <td>0</td>\n <td>1</td>\n </tr>\n </tbody>\n</table>\n</div>"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"train\_dataset = dataset.sample(frac=0.8,random\_state=0)\ntest\_dataset = dataset.drop(train\_dataset.index)\ntrain\_labels = train\_dataset['number\_of\_husbands']\ntest\_labels = test\_dataset['number\_of\_husbands']","execution\_count":132,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"print(train\_dataset)","execution\_count":133,"outputs":[{"output\_type":"stream","text":" Age number\_of\_husbands artist dancer doctor engineer lawyer \\\n18 19 29 0 1 0 0 0 \n1 24 3 0 1 0 0 0 \n19 30 30 0 0 0 0 0 \n8 30 20 0 0 0 0 0 \n10 24 24 1 0 0 0 0 \n17 30 28 0 0 0 0 0 \n6 28 13 0 0 0 1 0 \n13 26 19 0 0 0 0 1 \n4 24 10 0 0 0 0 0 \n2 38 2 1 0 0 0 0 \n5 36 8 0 0 0 1 0 \n14 30 28 0 0 0 0 1 \n9 20 26 0 0 0 0 0 \n7 29 15 0 0 1 0 0 \n16 29 20 0 0 0 1 0 \n11 40 20 0 1 0 0 0 \n\n painter singer \n18 0 0 \n1 0 0 \n19 0 1 \n8 1 0 \n10 0 0 \n17 1 0 \n6 0 0 \n13 0 0 \n4 0 1 \n2 0 0 \n5 0 0 \n14 0 0 \n9 0 1 \n7 0 0 \n16 0 0 \n11 0 0 \n","name":"stdout"}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"print(test\_dataset)","execution\_count":134,"outputs":[{"output\_type":"stream","text":" Age number\_of\_husbands artist dancer doctor engineer lawyer \\\n0 20 1 0 0 1 0 0 \n3 39 5 0 0 0 0 0 \n12 30 21 0 0 0 0 0 \n15 23 18 0 0 1 0 0 \n\n painter singer \n0 0 0 \n3 1 0 \n12 0 1 \n15 0 0 \n","name":"stdout"}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"print(train\_labels)","execution\_count":135,"outputs":[{"output\_type":"stream","text":"18 29\n1 3\n19 30\n8 20\n10 24\n17 28\n6 13\n13 19\n4 10\n2 2\n5 8\n14 28\n9 26\n7 15\n16 20\n11 20\nName: number\_of\_husbands, dtype: int64\n","name":"stdout"}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"print(test\_labels)","execution\_count":136,"outputs":[{"output\_type":"stream","text":"0 1\n3 5\n12 21\n15 18\nName: number\_of\_husbands, dtype: int64\n","name":"stdout"}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"def build\_model():\n model = keras.Sequential([layers.Dense(64, activation='relu',\n input\_shape=[len(train\_dataset.keys())]),layers.Dense(1)])\n #print(model)\n optimizer = tf.keras.optimizers.RMSprop(0.001)\n #print(optimizer)\n model.compile(loss='mse',\n optimizer=optimizer,\n metrics=['mae', 'mse'])\n return model\nmodel = build\_model()\n#print(model)","execution\_count":137,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"def build\_model():\n model = keras.Sequential([layers.Dense(64, activation='relu',\n input\_shape=[len(train\_dataset.keys())]),layers.Dense(1)])\n optimizer = tf.keras.optimizers.RMSprop(0.001)\n \n model.compile(loss='mse',\n optimizer=optimizer,\n metrics=['mae', 'mse'])\n return model\nmodel = build\_model()","execution\_count":138,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"import datetime\nlog\_dir=\"logs/fit/\" + datetime.datetime.now().strftime(\"%Y%m%d-%H%M%S\")","execution\_count":139,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"tensorboard\_callback = tf.keras.callbacks.TensorBoard(log\_dir=log\_dir, histogram\_freq=1)\nmodel.fit(x=train\_dataset, \n y=train\_labels, \n epochs=20, \n validation\_data=(test\_dataset, test\_labels), \n callbacks=[tensorboard\_callback])","execution\_count":140,"outputs":[{"output\_type":"stream","text":"Train on 16 samples, validate on 4 samples\nEpoch 1/20\n16/16 [==============================] - 1s 31ms/sample - loss: 549.5778 - mae: 20.6871 - mse: 549.5778 - val\_loss: 230.8543 - val\_mae: 11.3009 - val\_mse: 230.8543\nEpoch 2/20\n16/16 [==============================] - 0s 1ms/sample - loss: 511.6976 - mae: 19.8757 - mse: 511.6976 - val\_loss: 217.8123 - val\_mae: 10.9041 - val\_mse: 217.8123\nEpoch 3/20\n16/16 [==============================] - 0s 844us/sample - loss: 485.7766 - mae: 19.2981 - mse: 485.7766 - val\_loss: 207.6566 - val\_mae: 10.5784 - val\_mse: 207.6566\nEpoch 4/20\n16/16 [==============================] - 0s 874us/sample - loss: 465.1523 - mae: 18.8235 - mse: 465.1523 - val\_loss: 199.1346 - val\_mae: 10.4079 - val\_mse: 199.1346\nEpoch 5/20\n16/16 [==============================] - 0s 903us/sample - loss: 447.4073 - mae: 18.4257 - mse: 447.4073 - val\_loss: 191.3219 - val\_mae: 10.3776 - val\_mse: 191.3219\nEpoch 6/20\n16/16 [==============================] - 0s 921us/sample - loss: 431.5318 - mae: 18.0812 - mse: 431.5318 - val\_loss: 184.1292 - val\_mae: 10.3462 - val\_mse: 184.1292\nEpoch 7/20\n16/16 [==============================] - 0s 893us/sample - loss: 417.1359 - mae: 17.7607 - mse: 417.1359 - val\_loss: 177.4922 - val\_mae: 10.3112 - val\_mse: 177.4922\nEpoch 8/20\n16/16 [==============================] - 0s 863us/sample - loss: 403.7623 - mae: 17.4556 - mse: 403.7623 - val\_loss: 171.4126 - val\_mae: 10.2749 - val\_mse: 171.4126\nEpoch 9/20\n16/16 [==============================] - 0s 935us/sample - loss: 391.3177 - mae: 17.1651 - mse: 391.3177 - val\_loss: 165.7485 - val\_mae: 10.2392 - val\_mse: 165.7485\nEpoch 10/20\n16/16 [==============================] - 0s 911us/sample - loss: 379.5566 - mae: 16.8837 - mse: 379.5566 - val\_loss: 160.3589 - val\_mae: 10.1990 - val\_mse: 160.3589\nEpoch 11/20\n16/16 [==============================] - 0s 947us/sample - loss: 368.4781 - mae: 16.6133 - mse: 368.4781 - val\_loss: 155.2819 - val\_mae: 10.1564 - val\_mse: 155.2819\nEpoch 12/20\n16/16 [==============================] - 0s 946us/sample - loss: 358.0122 - mae: 16.3528 - mse: 358.0122 - val\_loss: 150.4872 - val\_mae: 10.1133 - val\_mse: 150.4872\nEpoch 13/20\n16/16 [==============================] - 0s 862us/sample - loss: 347.9652 - mae: 16.0976 - mse: 347.9652 - val\_loss: 145.8687 - val\_mae: 10.0690 - val\_mse: 145.8687\nEpoch 14/20\n16/16 [==============================] - 0s 956us/sample - loss: 338.2133 - mae: 15.8444 - mse: 338.2133 - val\_loss: 141.4783 - val\_mae: 10.0218 - val\_mse: 141.4783\nEpoch 15/20\n16/16 [==============================] - 0s 1ms/sample - loss: 328.8347 - mae: 15.5971 - mse: 328.8347 - val\_loss: 137.1779 - val\_mae: 9.9740 - val\_mse: 137.1779\nEpoch 16/20\n16/16 [==============================] - 0s 958us/sample - loss: 319.6334 - mae: 15.3654 - mse: 319.6334 - val\_loss: 133.0957 - val\_mae: 9.9259 - val\_mse: 133.0957\nEpoch 17/20\n16/16 [==============================] - 0s 1ms/sample - loss: 310.8037 - mae: 15.1581 - mse: 310.8037 - val\_loss: 129.2730 - val\_mae: 9.8827 - val\_mse: 129.2730\nEpoch 18/20\n16/16 [==============================] - 0s 1ms/sample - loss: 302.1890 - mae: 14.9534 - mse: 302.1890 - val\_loss: 125.5539 - val\_mae: 9.8340 - val\_mse: 125.5539\nEpoch 19/20\n16/16 [==============================] - 0s 919us/sample - loss: 293.7086 - mae: 14.7463 - mse: 293.7086 - val\_loss: 122.0458 - val\_mae: 9.7859 - val\_mse: 122.0458\nEpoch 20/20\n16/16 [==============================] - 0s 919us/sample - loss: 285.5493 - mae: 14.5429 - mse: 285.5493 - val\_loss: 118.7033 - val\_mae: 9.7393 - val\_mse: 118.7033\n","name":"stdout"},{"output\_type":"execute\_result","execution\_count":140,"data":{"text/plain":"<tensorflow.python.keras.callbacks.History at 0x7fb1a1149be0>"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"%load\_ext tensorboard","execution\_count":141,"outputs":[{"output\_type":"stream","text":"The tensorboard extension is already loaded. To reload it, use:\n %reload\_ext tensorboard\n","name":"stdout"}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"%tensorboard --logdir logs/fit","execution\_count":142,"outputs":[{"output\_type":"display\_data","data":{"text/plain":"Reusing TensorBoard on port 6006 (pid 59), started 1:55:02 ago. (Use '!kill 59' to kill it.)"},"metadata":{}},{"output\_type":"display\_data","data":{"text/plain":"<IPython.core.display.HTML object>","text/html":"\n <iframe id=\"tensorboard-frame-23d61dfb35d6a5a6\" width=\"100%\" height=\"800\" frameborder=\"0\">\n </iframe>\n <script>\n (function() {\n const frame = document.getElementById(\"tensorboard-frame-23d61dfb35d6a5a6\");\n const url = new URL(\"/\", window.location);\n url.port = 6006;\n frame.src = url;\n })();\n </script>\n "},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"ex=pd.DataFrame([[23,2,3,4,1,0,4,23,44]])","execution\_count":143,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"res=model.predict(ex)","execution\_count":144,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"print(res)","execution\_count":145,"outputs":[{"output\_type":"stream","text":"[[21.181519]]\n","name":"stdout"}]},{"metadata":{"trusted":true},"cell\_type":"code","source":" import matplotlib.pyplot as plt\n import numpy as np\n items\_count=np.array([1,2,3,4,5])\n price=np.array([5,10,15,20,25])\n plt.plot(items\_count,price,label=\"relation ship between count of objects and price\",color='b',marker='\*',linestyle='--')\n plt.legend()\n plt.xlabel(\"count\")\n plt.ylabel(\"price\")\n plt.show()","execution\_count":146,"outputs":[{"output\_type":"display\_data","data":{"text/plain":"<Figure size 432x288 with 1 Axes>","image/png":"\n"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"import matplotlib.pyplot as plt\nimport numpy as np\nitems\_count=np.array([1,2,3,4,5])\nprice=np.array([5,10,15,20,25])\nprice2=np.array([1,4,9,16,25])\nplt.plot(items\_count,price,label=\"object A\",color='b',marker='\*',linestyle='--')\nplt.plot(items\_count,price2,label=\"object B\",color='r',marker='.',linestyle='-')\nplt.legend()\nplt.xlabel(\"count\")\nplt.ylabel(\"price\")\nplt.show()","execution\_count":147,"outputs":[{"output\_type":"display\_data","data":{"text/plain":"<Figure size 432x288 with 1 Axes>","image/png":"\n"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"import numpy as np\nimport matplotlib.pyplot as plt\nyears=np.array([1975,1980,1985,1990,1995,2000])\nsales\_brandA=np.array([1000,2000,500,1500,2000,800])\nsales\_brandB=np.array([2000,800,400,2000,600,1000])\nfig,ax=plt.subplots()\nbar\_width=0.59\nplt.bar(years,sales\_brandA,label=\"A\",color='r',width=bar\_width)\nplt.bar(years+bar\_width,sales\_brandB,label=\"B\",color='b',width=bar\_width)\nplt.legend()\nplt.xlabel(\"years\")\nplt.ylabel(\"sales\")\nplt.show()","execution\_count":148,"outputs":[{"output\_type":"display\_data","data":{"text/plain":"<Figure size 432x288 with 1 Axes>","image/png":"\n"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"import numpy as np\nimport matplotlib.pyplot as plt\nyears=np.array([1975,1980,1985,1990,1995,2000])\nsales\_brandA=np.array([1000,2000,500,1500,2000,800])\nsales\_brandB=np.array([2000,800,400,2000,600,1000])\nfig,ax=plt.subplots()\nbar\_width=0.59\nplt.barh(years,sales\_brandA,label=\"A\",color='r')\nplt.barh(years+bar\_width,sales\_brandB,label=\"B\",color='b')\nplt.legend()\nplt.xlabel(\"years\")\nplt.ylabel(\"sales\")\nplt.show()","execution\_count":149,"outputs":[{"output\_type":"display\_data","data":{"text/plain":"<Figure size 432x288 with 1 Axes>","image/png":"\n"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"import pandas as pd\nimport matplotlib.pyplot as plt\n#Number of tourists visiting each city\ndata={\"CityA\":[20,30,40,50],\"CityB\":[30,70,60,50],\"CityC\":[10,80,60,100]}\nseasons={\"Spring\",\"Fall\",\"Summer\",\"Winter\"}\ndf=pd.DataFrame(data,index=seasons)\ndf.plot(kind='area',stacked=True)\nplt.show(block=True)","execution\_count":150,"outputs":[{"output\_type":"display\_data","data":{"text/plain":"<Figure size 432x288 with 1 Axes>","image/png":"\n"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"import matplotlib.pyplot as plt\n#Number of tourists visiting each city\ndata={\"CityA\":[20,30,40,50],\"CityB\":[30,70,60,50],\"CityC\":[10,80,60,100]}\nseasons={\"Spring\",\"Fall\",\"Summer\",\"Winter\"}\ndf=pd.DataFrame(data,index=seasons)\ndf.plot(kind='area',stacked=False)\nplt.show(block=True)","execution\_count":151,"outputs":[{"output\_type":"display\_data","data":{"text/plain":"<Figure size 432x288 with 1 Axes>","image/png":"\n"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"import numpy as np\nimport matplotlib.pyplot as plt\nfig,axes=plt.subplots()\nages=np.array([5,10,50,70,80,20,60])\nweights=np.array([8,20,89,60,45,70])\nplt.hist(ages,bins=20,label=\"ages\")\nplt.hist(weights,bins=20,label=\"weights\")\nplt.xlabel(\"quantity\")\nplt.ylabel(\"frequency\")\nplt.legend()\nplt.show()","execution\_count":152,"outputs":[{"output\_type":"display\_data","data":{"text/plain":"<Figure size 432x288 with 1 Axes>","image/png":"\n"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"import seaborn as sns","execution\_count":153,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"sns.set()","execution\_count":154,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":" import numpy as np\n import seaborn as sns\n import pandas as pd\n from sklearn import datasets\n sns.set()\n data=pd.read\_csv(\"C:/Users/admin/Downloads/iris.csv\")\n df1=pd.DataFrame(data)\n sns.lmplot(x='count',y='price',data=df1,hue=\"category\",fit\_reg=True)\n \n #in the scatter plot regression line is removed and colors are given category wise","execution\_count":166,"outputs":[{"output\_type":"error","ename":"FileNotFoundError","evalue":"[Errno 2] File b'C:/Users/admin/Downloads/iris.csv' does not exist: b'C:/Users/admin/Downloads/iris.csv'","traceback":["\u001b[0;31m---------------------------------------------------------------------------\u001b[0m","\u001b[0;31mFileNotFoundError\u001b[0m Traceback (most recent call last)","\u001b[0;32m<ipython-input-166-b74d97c29e75>\u001b[0m in \u001b[0;36m<module>\u001b[0;34m\u001b[0m\n\u001b[1;32m 4\u001b[0m \u001b[0;32mfrom\u001b[0m \u001b[0msklearn\u001b[0m \u001b[0;32mimport\u001b[0m \u001b[0mdatasets\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m 5\u001b[0m \u001b[0msns\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mset\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0;32m----> 6\u001b[0;31m \u001b[0mdata\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0mpd\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mread\_csv\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0;34m\"C:/Users/admin/Downloads/iris.csv\"\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0m\u001b[1;32m 7\u001b[0m \u001b[0mdf1\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0mpd\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mDataFrame\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mdata\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m 8\u001b[0m \u001b[0msns\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mlmplot\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mx\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0;34m'count'\u001b[0m\u001b[0;34m,\u001b[0m\u001b[0my\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0;34m'price'\u001b[0m\u001b[0;34m,\u001b[0m\u001b[0mdata\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0mdf1\u001b[0m\u001b[0;34m,\u001b[0m\u001b[0mhue\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0;34m\"category\"\u001b[0m\u001b[0;34m,\u001b[0m\u001b[0mfit\_reg\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0;32mTrue\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n","\u001b[0;32m/opt/conda/lib/python3.6/site-packages/pandas/io/parsers.py\u001b[0m in \u001b[0;36mparser\_f\u001b[0;34m(filepath\_or\_buffer, sep, delimiter, header, names, index\_col, usecols, squeeze, prefix, mangle\_dupe\_cols, dtype, engine, converters, true\_values, false\_values, skipinitialspace, skiprows, skipfooter, nrows, na\_values, keep\_default\_na, na\_filter, verbose, skip\_blank\_lines, parse\_dates, infer\_datetime\_format, keep\_date\_col, date\_parser, dayfirst, cache\_dates, iterator, chunksize, compression, thousands, decimal, lineterminator, quotechar, quoting, doublequote, escapechar, comment, encoding, dialect, error\_bad\_lines, warn\_bad\_lines, delim\_whitespace, low\_memory, memory\_map, float\_precision)\u001b[0m\n\u001b[1;32m 683\u001b[0m )\n\u001b[1;32m 684\u001b[0m \u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0;32m--> 685\u001b[0;31m \u001b[0;32mreturn\u001b[0m \u001b[0m\_read\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mfilepath\_or\_buffer\u001b[0m\u001b[0;34m,\u001b[0m \u001b[0mkwds\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0m\u001b[1;32m 686\u001b[0m \u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m 687\u001b[0m \u001b[0mparser\_f\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0m\_\_name\_\_\u001b[0m \u001b[0;34m=\u001b[0m \u001b[0mname\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n","\u001b[0;32m/opt/conda/lib/python3.6/site-packages/pandas/io/parsers.py\u001b[0m in \u001b[0;36m\_read\u001b[0;34m(filepath\_or\_buffer, kwds)\u001b[0m\n\u001b[1;32m 455\u001b[0m \u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m 456\u001b[0m \u001b[0;31m# Create the parser.\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0;32m--> 457\u001b[0;31m \u001b[0mparser\u001b[0m \u001b[0;34m=\u001b[0m \u001b[0mTextFileReader\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mfp\_or\_buf\u001b[0m\u001b[0;34m,\u001b[0m \u001b[0;34m\*\*\u001b[0m\u001b[0mkwds\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0m\u001b[1;32m 458\u001b[0m \u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m 459\u001b[0m \u001b[0;32mif\u001b[0m \u001b[0mchunksize\u001b[0m \u001b[0;32mor\u001b[0m \u001b[0miterator\u001b[0m\u001b[0;34m:\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n","\u001b[0;32m/opt/conda/lib/python3.6/site-packages/pandas/io/parsers.py\u001b[0m in \u001b[0;36m\_\_init\_\_\u001b[0;34m(self, f, engine, \*\*kwds)\u001b[0m\n\u001b[1;32m 893\u001b[0m \u001b[0mself\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0moptions\u001b[0m\u001b[0;34m[\u001b[0m\u001b[0;34m\"has\_index\_names\"\u001b[0m\u001b[0;34m]\u001b[0m \u001b[0;34m=\u001b[0m \u001b[0mkwds\u001b[0m\u001b[0;34m[\u001b[0m\u001b[0;34m\"has\_index\_names\"\u001b[0m\u001b[0;34m]\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m 894\u001b[0m \u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0;32m--> 895\u001b[0;31m \u001b[0mself\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0m\_make\_engine\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mself\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mengine\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0m\u001b[1;32m 896\u001b[0m \u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m 897\u001b[0m \u001b[0;32mdef\u001b[0m \u001b[0mclose\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mself\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m:\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n","\u001b[0;32m/opt/conda/lib/python3.6/site-packages/pandas/io/parsers.py\u001b[0m in \u001b[0;36m\_make\_engine\u001b[0;34m(self, engine)\u001b[0m\n\u001b[1;32m 1133\u001b[0m \u001b[0;32mdef\u001b[0m \u001b[0m\_make\_engine\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mself\u001b[0m\u001b[0;34m,\u001b[0m \u001b[0mengine\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0;34m\"c\"\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m:\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m 1134\u001b[0m \u001b[0;32mif\u001b[0m \u001b[0mengine\u001b[0m \u001b[0;34m==\u001b[0m \u001b[0;34m\"c\"\u001b[0m\u001b[0;34m:\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0;32m-> 1135\u001b[0;31m \u001b[0mself\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0m\_engine\u001b[0m \u001b[0;34m=\u001b[0m \u001b[0mCParserWrapper\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mself\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mf\u001b[0m\u001b[0;34m,\u001b[0m \u001b[0;34m\*\*\u001b[0m\u001b[0mself\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0moptions\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0m\u001b[1;32m 1136\u001b[0m \u001b[0;32melse\u001b[0m\u001b[0;34m:\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m 1137\u001b[0m \u001b[0;32mif\u001b[0m \u001b[0mengine\u001b[0m \u001b[0;34m==\u001b[0m \u001b[0;34m\"python\"\u001b[0m\u001b[0;34m:\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n","\u001b[0;32m/opt/conda/lib/python3.6/site-packages/pandas/io/parsers.py\u001b[0m in \u001b[0;36m\_\_init\_\_\u001b[0;34m(self, src, \*\*kwds)\u001b[0m\n\u001b[1;32m 1915\u001b[0m \u001b[0mkwds\u001b[0m\u001b[0;34m[\u001b[0m\u001b[0;34m\"usecols\"\u001b[0m\u001b[0;34m]\u001b[0m \u001b[0;34m=\u001b[0m \u001b[0mself\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0musecols\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m 1916\u001b[0m \u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0;32m-> 1917\u001b[0;31m \u001b[0mself\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0m\_reader\u001b[0m \u001b[0;34m=\u001b[0m \u001b[0mparsers\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mTextReader\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0msrc\u001b[0m\u001b[0;34m,\u001b[0m \u001b[0;34m\*\*\u001b[0m\u001b[0mkwds\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0m\u001b[1;32m 1918\u001b[0m \u001b[0mself\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0munnamed\_cols\u001b[0m \u001b[0;34m=\u001b[0m \u001b[0mself\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0m\_reader\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0munnamed\_cols\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m 1919\u001b[0m \u001b[0;34m\u001b[0m\u001b[0m\n","\u001b[0;32mpandas/\_libs/parsers.pyx\u001b[0m in \u001b[0;36mpandas.\_libs.parsers.TextReader.\_\_cinit\_\_\u001b[0;34m()\u001b[0m\n","\u001b[0;32mpandas/\_libs/parsers.pyx\u001b[0m in \u001b[0;36mpandas.\_libs.parsers.TextReader.\_setup\_parser\_source\u001b[0;34m()\u001b[0m\n","\u001b[0;31mFileNotFoundError\u001b[0m: [Errno 2] File b'C:/Users/admin/Downloads/iris.csv' does not exist: b'C:/Users/admin/Downloads/iris.csv'"]}]}],"metadata":{"kernelspec":{"language":"python","display\_name":"Python 3","name":"python3"},"language\_info":{"pygments\_lexer":"ipython3","nbconvert\_exporter":"python","version":"3.6.4","file\_extension":".py","codemirror\_mode":{"name":"ipython","version":3},"name":"python","mimetype":"text/x-python"}},"nbformat":4,"nbformat\_minor":4}