{

"nbformat": 4,

"nbformat\_minor": 0,

"metadata": {

"colab": {

"name": "K J S pavan kumar.ipynb",

"provenance": []

},

"kernelspec": {

"name": "python3",

"display\_name": "Python 3"

},

"language\_info": {

"name": "python"

}

},

"cells": [

{

"cell\_type": "code",

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 1000

},

"id": "PauhAANLF0jo",

"outputId": "7fc5435e-490b-4ca1-d7b9-d08ff4806971"

},

"source": [

"import numpy as np\n",

"import pandas as pd\n",

"import sklearn\n",

"from sklearn.datasets import load\_boston\n",

"df=load\_boston()\n",

"df.keys()\n",

"print(df.DESCR)\n",

"boston=pd.DataFrame(df.data,columns=df.feature\_names)\n",

"boston.head()\n",

"boston['MEDV']=df.target\n",

"boston.head()\n",

"boston.isnull()\n",

"boston.isnull.sum()"

],

"execution\_count": 2,

"outputs": [

{

"output\_type": "stream",

"text": [

".. \_boston\_dataset:\n",

"\n",

"Boston house prices dataset\n",

"---------------------------\n",

"\n",

"\*\*Data Set Characteristics:\*\* \n",

"\n",

" :Number of Instances: 506 \n",

"\n",

" :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.\n",

"\n",

" :Attribute Information (in order):\n",

" - CRIM per capita crime rate by town\n",

" - ZN proportion of residential land zoned for lots over 25,000 sq.ft.\n",

" - INDUS proportion of non-retail business acres per town\n",

" - CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)\n",

" - NOX nitric oxides concentration (parts per 10 million)\n",

" - RM average number of rooms per dwelling\n",

" - AGE proportion of owner-occupied units built prior to 1940\n",

" - DIS weighted distances to five Boston employment centres\n",

" - RAD index of accessibility to radial highways\n",

" - TAX full-value property-tax rate per $10,000\n",

" - PTRATIO pupil-teacher ratio by town\n",

" - B 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town\n",

" - LSTAT % lower status of the population\n",

" - MEDV Median value of owner-occupied homes in $1000's\n",

"\n",

" :Missing Attribute Values: None\n",

"\n",

" :Creator: Harrison, D. and Rubinfeld, D.L.\n",

"\n",

"This is a copy of UCI ML housing dataset.\n",

"https://archive.ics.uci.edu/ml/machine-learning-databases/housing/\n",

"\n",

"\n",

"This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University.\n",

"\n",

"The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic\n",

"prices and the demand for clean air', J. Environ. Economics & Management,\n",

"vol.5, 81-102, 1978. Used in Belsley, Kuh & Welsch, 'Regression diagnostics\n",

"...', Wiley, 1980. N.B. Various transformations are used in the table on\n",

"pages 244-261 of the latter.\n",

"\n",

"The Boston house-price data has been used in many machine learning papers that address regression\n",

"problems. \n",

" \n",

".. topic:: References\n",

"\n",

" - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data and Sources of Collinearity', Wiley, 1980. 244-261.\n",

" - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings on the Tenth International Conference of Machine Learning, 236-243, University of Massachusetts, Amherst. Morgan Kaufmann.\n",

"\n"

],

"name": "stdout"

},

{

"output\_type": "error",

"ename": "AttributeError",

"evalue": "ignored",

"traceback": [

"\u001b[0;31m---------------------------------------------------------------------------\u001b[0m",

"\u001b[0;31mAttributeError\u001b[0m Traceback (most recent call last)",

"\u001b[0;32m<ipython-input-2-bdad6c0c4758>\u001b[0m in \u001b[0;36m<module>\u001b[0;34m()\u001b[0m\n\u001b[1;32m 11\u001b[0m \u001b[0mboston\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mhead\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m 12\u001b[0m \u001b[0mboston\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0misnull\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0;32m---> 13\u001b[0;31m \u001b[0mboston\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0misnull\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0msum\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0m",

"\u001b[0;31mAttributeError\u001b[0m: 'function' object has no attribute 'sum'"

]

}

]

},

{

"cell\_type": "code",

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "TBK6bHxFIeNQ",

"outputId": "85995871-8ac0-4156-8b97-ec57583c470a"

},

"source": [

"from sklearn.model\_selection import train\_test\_split\n",

"x=boston.drop(\"MEDV\",axis=1)\n",

"y=boston[\"MEDV\"]\n",

"x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.15,random\_state=5)\n",

"print(x\_train.shape)\n",

"print(x\_test.shape)\n",

"print(y\_train.shape)\n",

"print(y\_test.shape)"

],

"execution\_count": 3,

"outputs": [

{

"output\_type": "stream",

"text": [

"(430, 13)\n",

"(76, 13)\n",

"(430,)\n",

"(76,)\n"

],

"name": "stdout"

}

]

},

{

"cell\_type": "code",

"metadata": {

"id": "NR21Z7VuJ7LA"

},

"source": [

"from sklearn.linear\_model import LinearRegression\n",

"from sklearn.metrics import mean\_squared\_error"

],

"execution\_count": 7,

"outputs": []

},

{

"cell\_type": "code",

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "KRJSGMCUKjoe",

"outputId": "4c41fe26-5880-4cfc-d0d4-6f6e6cf34c2e"

},

"source": [

"lin\_model=LinearRegression()\n",

"lin\_model.fit(x\_train,y\_train)"

],

"execution\_count": 8,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=None, normalize=False)"

]

},

"metadata": {

"tags": []

},

"execution\_count": 8

}

]

},

{

"cell\_type": "code",

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "i98ADHb7KyqV",

"outputId": "08337512-2267-4024-dc04-987d54138855"

},

"source": [

"y\_train\_predict=lin\_model.predict(x\_train)\n",

"rmse=(np.sqrt(mean\_squared\_error(y\_train,y\_train\_predict)))\n",

"\n",

"print(\"the model performance for training set\")\n",

"print('RMSE is {}'.format(rmse))\n",

"print(\"\\n\")\n",

"y\_test\_predict=lin\_model.predict(x\_test)\n",

"rmse=(np.sqrt(mean\_squared\_error(y\_test,y\_test\_predict)))\n",

"print(\"The model performance for testing set\")\n",

"print('RMSE is {}'.format(rmse))"

],

"execution\_count": 10,

"outputs": [

{

"output\_type": "stream",

"text": [

"the model performance for training set\n",

"RMSE is 4.710901797319795\n",

"\n",

"\n",

"The model performance for testing set\n",

"RMSE is 4.687543527902971\n"

],

"name": "stdout"

}

]

}

]

}